MASTER EN SCIENCES INFORMATIQUES

Couplage d'un glossaire avec un système d'apprentissage multimédia

Lecerf, Audrey

Award date:
2001

Awarding institution:
Universite de Namur

FACULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX
INSTITUT D'INFORMATIQUE
NAMUR

Couplage d'un glossaire avec un système d'apprentissage multimédia

Audrey LECERF

Mémoire présenté en vue de l'obtention
du grade de Maître en Informatique.

Année académique 2000-2001

Résumé

Ce mémoire retrace les étapes d'analyse, de spécification et de conception d'une application créée dans le cadre de la troisième maîtrise en informatique aux FUNDP. Cette application constitue la partie glossaire d'un système d'apprentissage multimédia, objet du projet MAI de l'Institut d'Informatique. Le système MAI est un système auteur. D'une part, il permet à des professeurs de construire des cours composés de modules interactifs. D'autre part, des apprenants peuvent consulter ces cours selon un parcours représenté par une carte de navigation. L'outil créé dans le contexte de ce travail consiste à aider un professeur à créer un nouveau glossaire, importer un glossaire existant ou bien modifier un glossaire en supprimant ou insérant des termes. L'apprenant peut, quant à lui, obtenir, grâce au programme réalisé, des informations de tout genre au sujet d'un terme inconnu ou compliqué rencontré dans le cours qu'il consulte.

Abstract

This thesis presents the stages of analysis, specification and conception of an application that has been created to obtain a master's degree in Computer Science at the FUNDP. That application represents the glossary of a multimedia learning system which is under construction within the framework of the MAI project of the Computer Science Institute. The MAI system is an authoring system that allows teachers to create courses consisting of interactive teaching modules. Learners can consult these courses by following a path schematized by a map. The function of the tool created for this work is to assist a teacher in building a new glossary, in importing an existing glossary and in modifying a glossary by deleting or by adding some terms. Thanks to the tool that has been produced, the learner can obtain further information about an unknown or complex term that he encounters while consulting a course.

Arrivée au terme de ce travail, je tiens à témoigner toute ma gratitude à mon promoteur, le professeur Monique NOIRHOMME, ainsi qu'à mes collègues, Didier VAISER, Isabelle HOUSEN, Philippe CALMANT et surtout Maurice NDAYE MUKUNA pour m'avoir prodigué de nombreux conseils et m'avoir guidée tout au long de cette entreprise.

Je désire ensuite remercier tout particulièrement mon amie Catherine qui a consacré énormément de temps à m'aider à réaliser un mémoire de fin d'études qui soit le plus respectueux possible de notre belle langue française.

Merci encore à Lucie, Stéphane et Cédric, pour leur amitié et leur disponibilité, à Denis, mon maître de stage à Exeter, pour son dévouement, et à Clare, Becky, Carol et Mervyn, mes amis anglais, qui m'ont rendu le sourire lorsque j'avais le mal du pays.

Je remercie également ma sœur Ariane, mon fiancé Dimitri, ainsi que mes parents, pour leur patience et leurs encouragements.

Mes remerciements s'adressent enfin à tous ceux qui ont contribué, de près ou de loin, à l'aboutissement de ce travail.

Table des matières

Introduction

Dans toute œuvre littéraire ou scientifique, il s'avère qu'un glossaire joue un rôle essentiel. En effet, qui ne s'est pas retrouvé, un jour, interloqué devant un terme inconnu - ou spécifique au contexte - lui interdisant de comprendre le sens d'une phrase, ou même d'un paragraphe entier ? La présence d'un glossaire, au début ou en fin d'ouvrage, constitue dès lors une aide précieuse. Ces dernières années, l'apparition du World Wide Web a permis à de nombreux auteurs de faire circuler leurs publications à l'échelle internationale. Ainsi, bon nombre de documents pédagogiques et de systèmes d'apprentissage sont accessibles à partir d'Internet. Un glossaire constituant, à nos yeux, un outil indispensable à tout personne en quête de savoir, il nous semble souhaitable que ces ouvrages, sous leur forme numérique, possèdent eux aussi une partie glossaire. En 1999, les Facultés Universitaires Notre-Dame de la Paix (FUNDP) ont mis sur pied un projet de création d'un système d'apprentissage multimédia destiné à fonctionner via le Web : le système auteur MAI ("Modules d'Apprentissage Interactifs"). Ce système comporte deux aspects : le professeur et l'apprenant. Le professeur crée des cours que l'apprenant consulte. Ce mémoire est consacré à la réalisation de l'outil glossaire de ce système auteur et aborde toutes les étapes de sa création. Il s'agit d'un outil de présentation du glossaire d'un cours (destiné aux apprenants) mais aussi de création d'un glossaire (destiné aux professeurs).

Afin de nous familiariser avec le développement d'applications Web et avec le langage de programmation Java, nous avons effectué un stage au Devon County Council d'Exeter, en Angleterre. Le premier chapitre de ce mémoire fait fonction de rapport de stage. Il présente le contexte dans lequel s'est déroulé le stage, une description du programme conçu lors du stage, et, plus particulièrement, ses originalités, comme, par exemple, les métadonnées et l'accès à une base de données à partir d'une applet.

Le deuxième chapitre de cet exposé constitue une réflexion sur la définition d'un glossaire pédagogique, sur son utilisation, sur sa présentation et sur son contenu. Ces réflexions ont été alimentées par des recherches en bibliothèque et sur Internet.

Les résultats de ces investigations, confrontés avec les idées proposées, avant et après le stage, par les membres de l'équipe du projet MAI, ont servi de base aux spécifications exposées dans le quatrième chapitre.

Comme nous venons de l'annoncer, ce mémoire a été rédigé dans le cadre du projet MAI, né d'une collaboration entre l'Institut d'Informatique et le Département de Biologie de la Faculté des Sciences des FUNDP. En effet, l'Institut d'Informatique a mis sur pied ce projet en vue de concevoir un système auteur permettant à un professeur de créer un cours multimédia qui peut être, par la suite, consulté par des apprenants. Chaque cours se compose de plusieurs unités d'apprentissage qui sont organisées en réseau sémantique. Ce dernier, matérialisé par une carte de navigation, permet à l'apprenant de parcourir le cours. Dans ce projet, le Département de Biologie de la Faculté des Sciences a pour mission de réaliser le contenu du système auteur, c'est-à-dire le prototype d'un cours de génie génétique moléculaire, en vue d'aider à la définition des besoins et à la validation de l'outil créé par l'équipe d'informaticiens. Le troisième chapitre du mémoire décrit de manière détaillée toutes les caractéristiques du système MAI.

Comme la plupart des documents pédagogiques, le système MAI offre à l'utilisateur la possibilité de consulter un glossaire afin d'obtenir des informations précises sur un terme inconnu ou mal compris. Chaque cours répertorié dans le système MAI possède son glossaire. Celui-ci peut être consulté par un apprenant. Le professeur a, quant à lui, la possibilité de créer un nouveau glossaire, d'en importer un ou bien d'en modifier le contenu, en y ajoutant ou supprimant des termes. L'utilisateur peut accéder au glossaire d'un cours directement à partir de l'interface du système, mais aussi, indirectement, à partir d'une unité d'apprentissage, grâce à des liens les unissant. Toutes ces fonctionnalités sont développées dans l'application faisant l'objet de ce mémoire. Les étapes d'analyse, de spécification et de conception de cette application, ainsi que la base de données associée, sont présentées dans le quatrième chapitre.

Finalement, le lecteur trouvera, en annexe A du mémoire, des captures d'écran des glossaires "on-line" étudiés. L'annexe B présente les interfaces de l'application réalisée dans le cadre du mémoire. Ensuite, désirant échapper au dicton selon lequel "le cordonnier est toujours le plus mal chaussé", nous proposons au lecteur un glossaire des termes techniques du mémoire dans l'annexe C. Un exemple d'unité d'apprentissage réalisée par l'équipe de biologistes se trouve en annexe D. Les annexes E et F contiennent respectivement le schéma conceptuel et le schéma logique relationnel de la base de données du système MAI. Le script SQL de création de la partie de la base de données MAI consacrée au glossaire figure en annexe G. Enfin, le lecteur trouvera, en annexe H, le code Java correspondant au glossaire implémenté dans le contexte du mémoire.

1
Stage au Devon County Council

Du 8 septembre au 22 décembre 2000, nous avons effectué un stage en entreprise, en Angleterre, dans le cadre de la troisième maîtrise en informatique aux FUNDP. Le travail demandé consistait à développer une application Web, en langage Java, au Devon County Council[1]. Ce chapitre débute par une présentation du lieu de stage et poursuit par une description détaillée du travail réalisé au Devon County Council. Nous y développons plus particulièrement deux points originaux de ce travail, à savoir l'utilisation de métadonnées et l'accès à une base de données à partir d'une applet.

1.1. Présentation du Devon County Council

Le Devon County Council est situé dans la merveilleuse ville d'Exeter, en Grande-Bretagne. Exeter est la capitale et le centre administratif du Devon, l'un des trois plus grands comtés de Grande-Bretagne. Il compte plus d'un million d'habitants et Exeter totalise, à elle seule, plus de 100 000 habitants.

Le Devon County Council, fondé en 1888, est le gouvernement local du comté du Devon. Il s'agit de la plus grande autorité du sud-ouest de l'Angleterre. Celui-ci offre de nombreux services à une population sans cesse croissante et travaille en partenariat avec les autorités locales, les secteurs privés et publics, les associations de commerçants et la population locale.

[1] Voir : http://www.devon.gov.uk

1.1.1. Les services offerts à la population du Devon

Les services proposés aux citoyens par le Devon County Council comprennent, notamment :

- 373 écoles assurant l'éducation de 93 000 enfants ;
- 67 bibliothèques ;
- la maintenance de 12 000 km de routes ;
- une aide sociale à plus de 12 000 adultes ;
- la protection de près de 800 enfants défavorisés.

1.1.2. La politique du Devon County Council

Le Devon County Council a adopté six objectifs qui sous-tendent chaque décision :

- pourvoir aux besoins actuels sans compromettre ceux des générations futures ;
- développer et éduquer la population ;
- promouvoir la santé et la sécurité dans le Devon ;
- protéger et prendre soin des habitants du comté ;
- protéger et mettre en valeur l'environnement du Devon ;
- travailler pour la prospérité et contre la pauvreté.

Sur le site Web, Brian Greenslade, leader du Devon County Council, explique la politique de ce dernier en ces termes :

> *"We are committed to providing good quality and cost effective services to the people of Devon and we do this by identifying individual and community needs, and by working in partnership with others to achieve the best results."*
>
> *"Above all we want to involve the people of Devon in more decision making because it leads to better informed policies and services which are more responsive to local needs."*

1.1.3. La structure du Devon County Council

Le Devon County Council se décompose en cinq grands départements :

- la direction générale et ses conseillers ;
- la culture[2] ;
- l'environnement ;
- les ressources ;
- les services sociaux.

1.2. Stage au Devon County Council

1.2.1. Description de l'existant

Le stage s'est déroulé dans le département "Ressources" du Devon County Council. Il a consisté en une collaboration avec une équipe de statisticiens et de mathématiciens. Cette équipe gère toutes les données collectées, depuis 1980, dans tout le comté du Devon. Ces informations ont trait à plus de 80 domaines différents, tels que : l'agriculture, les tendances démographiques, les statistiques électorales, l'environnement, les accidents de la route, le tourisme, le chômage... Toutes les données s'y rapportant sont stockées dans des feuilles de calcul mises à jour quotidiennement.

Actuellement, les feuilles de calcul ainsi que certains graphes illustrant l'évolution des données dans le temps peuvent être visionnés dans la partie "Facts and Figures"[3] du site Web du Devon County Council. Les différentes feuilles de calcul sont pertinemment regroupées par sujet mais elles ne présentent aucune structure bien établie et n'offrent pas de grande facilité de navigation ni de comparaison à l'utilisateur.

---

[2] La culture regroupe l'éducation, les arts et les bibliothèques.
[3] Voir : http://www.devon.gov.uk/dris/homepage.html

1.2.2. La restructuration

Le travail effectué pour le Devon County Council s'est déroulé dans le cadre d'un projet de restructuration de la partie "Facts and Figures" du site Web. Cette restructuration, qui n'en était, alors, qu'à ses débuts, cherchait à ce que l'utilisateur du site puisse visionner des intervalles de données restreints à une région géographique particulière et/ou à un intervalle de temps précis. Elle visait également la possibilité de comparer les données à propos d'un même sujet, mais pour des régions géographiques différentes ou des périodes distinctes. Dans un premier temps, ces comparaisons seraient facilitées par la présentation des données sous la forme de tableaux et, par la suite, sous la forme de graphiques.

1.2.3. Le stage

1.2.3.1. Description sommaire du système dans son ensemble et perspectives

Au cours de ce stage, il nous a été demandé de créer un prototype qui serait utilisé uniquement à des fins de démonstration. Une fois réalisé, ce prototype serait, en premier lieu, évalué en vue de vérifier si les spécifications énoncées ont été respectées et s'il répond effectivement aux besoins des futurs utilisateurs. Par la suite, le Devon County Council engagerait de nouveaux programmeurs qui recevraient pour mission le perfectionnement du système ébauché, notamment en lui ajoutant de nouvelles fonctions. Lors de la troisième étape, l'interface utilisateur du prototype serait modifiée, de manière à l'harmoniser avec le design préexistant du site Web. Cette étape terminée, il serait alors opportun de publier le nouveau système sur le réseau interne du Devon County Council. Dans un premier temps, seul le personnel du Devon County Council pourrait, dès lors, l'utiliser pour faciliter et accélérer son travail quotidien. Chaque membre du personnel aurait ainsi l'occasion d'évaluer le système. Suite à d'éventuelles remarques, une dernière version du produit serait publiée sur le réseau international.

1.2.3.2. Description détaillée de la partie du système concernée par le stage

Comme nous l'avons dit précédemment, le système à créer avait pour objectif de permettre à son utilisateur de visualiser des statistiques, sous la forme d'un tableau, via Internet. Pour pouvoir fonctionner sur le net, le système devait être incorporé à une page HTML. C'est pourquoi notre choix s'est porté sur l'objet Applet du langage de programmation Java. Selon [VOSS 98], une applet est "un programme Java, intégré tel quel dans une page HTML, et qui accomplit une fonction. Ce programme tourne quelle que soit la plate-forme qui le supporte". Afin d'optimiser le rangement des statistiques ainsi que leur accès, nous avons proposé de transférer les données des feuilles de calcul vers une base de données. Une base de données prototype a donc été créée par l'un de nos collaborateurs, au moyen de Microsoft Access 97. Cette base de données ne contenait qu'un minuscule sous-ensemble des données que possède le Devon County Council.

a) Les métadonnées

La base de données à laquelle doit accéder l'applet possède deux types de tables. Les tables du premier type contiennent des métadonnées. Les métadonnées sont des informations concernant les données collectées[4]. Ces dernières font partie, quant à elles, du second type de tables. Elles représentent les véritables données intéressant l'utilisateur.

Il semble utile de prendre connaissance des informations identifiant une donnée du Devon County Council :

- une donnée se rapporte toujours à un sujet bien précis parmi un ensemble de plus de 80 thèmes extrêmement divers (population, chômage, handicaps...) ;
- elle est collectée à un certain niveau géographique (paroisse, circonscription électorale, district, comté...) ;
- elle peut être disponible pour un certain nombre de régions appartenant à ce niveau géographique mais pas toujours pour toutes ;

[4] Dans notre système, l'ensemble des données collectées se limite aux dix sujets faisant partie du prototype.

- elle peut avoir été collectée mensuellement, trimestriellement ou annuellement.

Ces quatre informations constituent les métadonnées décrivant les données collectées par le Devon County Council et demandées par l'utilisateur. A ce stade de l'exposé, il nous semble intéressant d'approfondir notre approche de chacune de ces métadonnées :

☙ *Le sujet*

- Chaque sujet caractérisant une donnée peut faire partie d'un groupe de sujets.
- Chaque groupe de sujets peut contenir un autre groupe de sujets, un sujet ou un document.
- Un document est une page HTML se rapportant au groupe de sujets sélectionné et est identifié par une URL.

☙ *Le niveau géographique*

- Chaque niveau géographique se place dans une hiérarchie de niveaux structurée comme un arbre : elle possède une racine (niveau supérieur), des branches (niveaux intermédiaires) et des feuilles (niveau inférieur).

☙ *La région*

- Chaque région dans laquelle les données sont collectées appartient à un seul niveau géographique existant. Par exemple, la région portant le nom "Exeter" correspond au niveau géographique appelé "District" (district), alors que la région nommée "St David's" référence le niveau géographique "Ward" (circonscription électorale).

☙ *La période*

- Les périodes sont classées dans une liste contenant d'abord les mois (janvier 1990 à décembre 2000), ensuite les années (année 1990 à année 2000) et enfin les trimestres (trimestre 1 en 1997 à trimestre 4 en 2000).

De longues réflexions et bon nombre de comparaisons entre différentes structures de données existantes ont abouti à l'adoption de deux représentations distinctes pour ces métadonnées : les hiérarchies et les listes liées. Ces métadonnées peuvent, par conséquent, être classées dans deux catégories, selon la structure choisie pour les représenter : les hiérarchies et les listes liées.

☙ *Les hiérarchies*

Les notions de "sujet", "niveau géographique" et "région" peuvent être représentées de manière similaire par des arbres ou "hiérarchies". Selon les spécifications du projet, chaque nœud d'un arbre ne peut avoir qu'un seul parent mais peut avoir plusieurs fils. Dans cette structure hiérarchique, chaque nœud est concrétisé par un objet qui possède une référence vers son père (qui est un autre objet) et une référence vers une liste liée contenant l'ensemble de ses fils. La notion de liste liée est expliquée dans le point suivant. Il existe néanmoins une différence importante entre la structure hiérarchique représentant la métadonnée "sujet" et celle représentant les métadonnées "niveau géographique" et "région". En effet, alors que l'arbre des niveaux géographiques et des régions ne peut posséder qu'une seule racine, comme l'illustrent les figures 1.1 et 1.2, il peut exister plusieurs racines dans la hiérarchie des groupes de sujets, sujets et documents. La figure 1.3 illustre une telle hiérarchie.

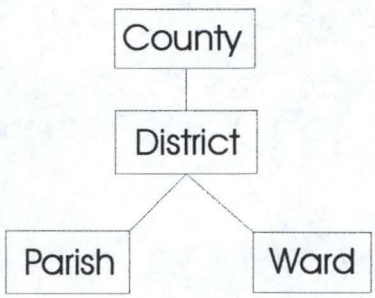


Figure 1.1 – Hiérarchie des niveaux géographiques

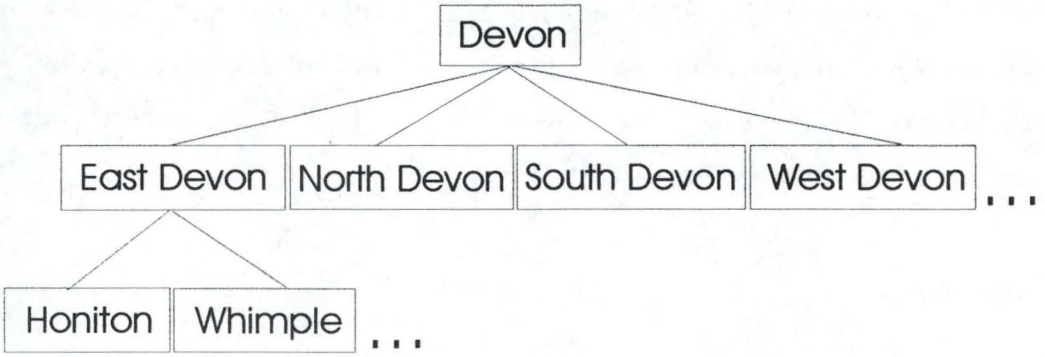


Figure 1.2 – Hiérarchie des régions

Ces hiérarchies permettent de structurer les métadonnées. Elles sont créées lors de la première étape du fonctionnement du système. Celle-ci consiste à accéder aux tables de la base de données contenant les métadonnées : les "métatables". Pour chaque enregistrement de chaque "métatable"[5], le système crée un nouvel objet ainsi qu'une référence éventuelle vers son parent. Bien entendu, ce n'est pas le cas pour l'(ou les) objet(s) racine(s) qui n'a(ont) pas de parent. Au même moment, s'il existe, le parent se voit attribuer une référence vers son nouveau fils. Par conséquent, la hiérarchie croît au fur et à mesure qu'un nouvel objet est créé à partir d'un enregistrement d'une "métatable".

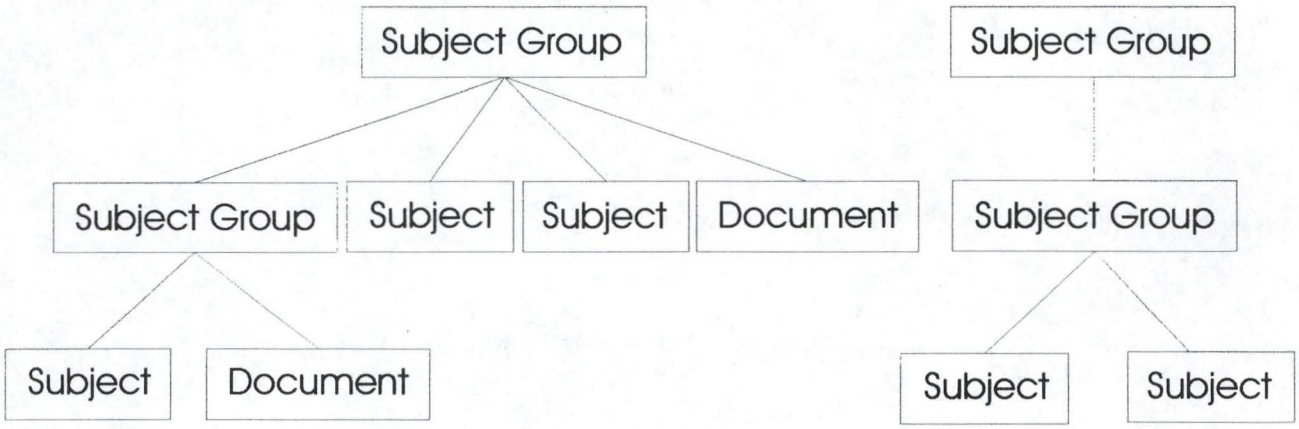


Figure 1.3 – Hiérarchie des groupes de sujets, sujets et documents

[5] C'est-à-dire pour chaque métadonnée.

Les listes liées

La métadonnée "période" est mise en œuvre par une disposition en liste liée, décrite dans [LEMAY-CADENHEAD 00] comme une structure de données servant à stocker une liste d'objets. L'accès à ces objets s'y fait de manière séquentielle, contrairement à l'accès par indice numérique de la plupart des structures de données standards de Java. Il est donc impossible d'accéder de manière aléatoire aux éléments d'une liste liée. Le principal avantage de cette structure est la très grande facilité d'ajout, de suppression et d'insertion d'éléments.

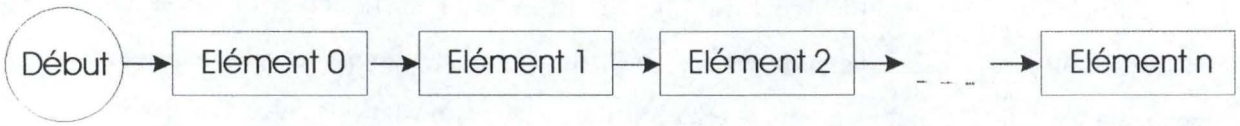


Figure 1.4 – Liste liée simple

La figure 1.4 représente une liste liée simple identique à celle utilisée par le système créé lors du stage. Cette structure de données n'est pas uniquement utilisée dans le cadre de la génération des objets "période", mais aussi lors de la création de chaque nouveau fils d'un nœud d'une hiérarchie. En effet, comme nous l'avons énoncé ci-dessus, chaque objet d'une hiérarchie possède une référence vers son père mais aussi une référence vers la liste de ses fils. L'arrivée d'un nouveau fils se traduit par l'ajout de celui-ci à la fin de la liste liée référencée par le père.

Une autre utilisation de la liste liée, dans ce système, concerne les différents groupes de sujets qui jouent le rôle de racines de l'arborescence des groupes de sujets, sujets et documents. De fait, ces objets "racines" sont regroupés dans une liste liée.

Les différentes classes d'objets du système représentant les métadonnées sont :

- GeographicLevel;
- GeographicArea ;
- SubjectGroup ;
- Subject ;
- Document ;
- Period.

Par ailleurs, il existe une classe Hierarchy dont le rôle est de centraliser toutes les références vers les métadonnées. Un objet Hierarchy possède une référence vers chaque racine de hiérarchie différente, ainsi que vers la liste des périodes. De leur côté,

tous les objets jouant le rôle de racines d'arborescences détiennent une référence vers l'objet Hierarchy. Cet objet est unique. Il est créé au début de l'étape de traitement et chargement des métadonnées. Un objet Hierarchy est donc identifié par :

- le niveau géographique racine de l'arborescence des niveaux géographiques ;
- la région géographique racine de l'arborescence des régions géographiques ;
- la liste liée des groupes de sujets racines de l'arborescence des sujets ;
- la liste liée de toutes les périodes.

Les métadonnées et leur structuration sous forme de hiérarchies et de listes liées sont très utiles. Elles permettent en effet un gain de temps et de meilleures performances de l'applet pendant son exécution et ses interactions avec l'utilisateur. L'accès à la base de données et la création des objets et références entre objets sont exécutés pendant le chargement de l'applet. Le lancement de cette applet est donc assez lent comparativement à un chargement ordinaire. Privilégier le temps d'exécution au détriment du temps de chargement est un choix tout à fait subjectif et découle de la décision d'utiliser des métadonnées. Afin de mieux comprendre la rapidité des interactions entre le système et l'utilisateur, il nous semble utile d'introduire une brève description de ces interactions (qui sont parfaitement séquentielles) :

1) L'utilisateur choisit un sujet[6] ou un document qui l'intéresse parmi ceux figurant dans une arborescence de groupes de sujets, sujets et documents ;
2) L'utilisateur choisit un niveau géographique parmi ceux figurant dans une arborescence de niveaux géographiques ;
3) L'utilisateur choisit une ou plusieurs régions géographiques parmi celles figurant dans une arborescence de régions ;
4) L'utilisateur choisit une période dans une liste de choix contenant les périodes pour lesquelles des données sont disponibles ;
5) Le système affiche un tableau reprenant les données concernant le sujet sélectionné par l'utilisateur, collectées au niveau géographique demandé, pour les régions choisies et dans l'intervalle de temps désiré.

---

[6] Remarquons que l'utilisateur est obligé de sélectionner une feuille de l'arborescence, correspondant à un sujet ou un document. Il lui est interdit de choisir l'un des nœuds de l'arbre qui représentent les groupes de sujets pour lesquels aucune donnée n'a été collectée.

L'utilisateur est obligé de respecter l'ordre des interactions car un nouveau panneau contenant l'arborescence (ou la liste de choix) adéquate apparaît à la fin de chaque interaction. En réalité, avant l'affichage de chaque nouveau panneau, l'arborescence ou liste de choix suivante est mise à jour en fonction des choix précédents de l'utilisateur. Cette mise à jour est accélérée par le fait que tous les objets devant apparaître dans les arborescences ou liste de l'interface utilisateur existent déjà. De fait, elles ont été créés lors du lancement de l'applet. Ces objets sont structurés en hiérarchies ou listes et possèdent des références vers leur parent, leurs fils et/ou leurs frères. La mise à jour des arborescences dans l'interface utilisateur est réalisée par un simple parcours des arbres (père → fils ou frère) ou des listes d'objets (frère 1 → frère 2). Aucun nouvel accès à la base de données n'est réalisé pendant les quatre premières interactions entre l'utilisateur et le système. Une nouvelle connexion à la base de données est ouverte lors de la dernière interaction, et ce, afin d'accéder aux données respectant toutes les conditions requises par l'utilisateur. Ces données sont ensuite affichées dans un grand tableau.

Pour résumer le fonctionnement du système, nous pouvons dire qu'il commence par accéder aux "métatables" de la base de données, dans le but de créer un nouvel objet pour chaque enregistrement de ces tables. Tous ces objets se référencent les uns les autres au moyen de structures de données en arbres ou en listes liées. La racine des arbres représentant les niveaux géographiques et les régions, la liste des racines de la hiérarchie contenant les groupes de sujets, ainsi que la liste des périodes, sont référencés par l'unique objet Hierarchy créé au début de cette première étape. A partir de cet objet Hierarchy, il est possible de parcourir toutes les hiérarchies et listes liées créées pour connaître tous les objets nécessaires à la réalisation des arborescences et de la liste de choix qui apparaîtront dans l'interface, au fur et à mesure que l'utilisateur progressera dans ses choix. Quand tous les choix ont été déterminés, une deuxième et dernière connexion à la base de données permet d'afficher dans un tableau toutes les données satisfaisant aux conditions prescrites par l'utilisateur.

b) Accès à la base de données au moyen du langage de programmation Java

Java permet à des applications ou à des applets de se connecter à une base de données au moyen de la technologie JDBC ("Java DataBase Connectivity"). Selon [LEMAY-CADENHEAD 00], la technologie JDBC est une bibliothèque de classes

offrant la possibilité aux programmes Java de travailler avec une grande diversité de formats de bases de données relationnelles. En effet, chaque système de gestion de bases de données utilise ses propres techniques, qualifiées de "propriétaires", pour accéder aux données. Le JDBC utilise un langage de requêtes standard : SQL ("Structured Query Language") qui simplifie l'utilisation des systèmes de gestion de bases de données. SQL est un langage de requête offrant des avantages imparables : portabilité des applications, indépendance par rapport au SGBD ("Systèmes de Gestion de Bases de Données"), stabilité des formations... Une requête SQL est une opération qui consiste à demander des informations stockées dans une base de données. Ces informations sont renvoyées par le système de gestion de la base de données sous la forme d'une liste d'enregistrements que l'on peut lire de manière séquentielle. La bibliothèque de classes de la technologie JDBC contient des classes exécutant des tâches telles que :

- l'établissement d'une connexion avec une base de données ;
- la création d'une instruction en SQL ;
- l'exécution d'une requête SQL sur une base de données ;
- la visualisation des enregistrements résultant de la requête SQL.

Grâce au langage de requêtes standard SQL, un programme Java peut interagir avec une base de données quels que soient son format et la plate-forme pour laquelle elle a été créée. Cette indépendance est assurée par un gestionnaire de pilotes. En effet, à chaque format de base de données correspond un pilote différent. Parmi ceux-ci, il existe des pilotes de bases de données JDBC permettant de relier des applications Java à des bibliothèques d'accès à des bases de données. Ces pilotes peuvent être écrits entièrement en Java ou bien implémentés au moyen de code natif.

☙ *Le cas des applications Java : Le pont JDBC–ODBC*

Une application Java est un programme Java autonome qui ne nécessite pas de navigateur Web pour fonctionner. Elle se distingue d'une applet Java par son mode d'exécution. L'exécution d'une application Java nécessite un interpréteur Java qui charge le fichier ".class" principal de l'application. Une application Java se caractérise également par un accès illimité à l'ensemble des composants du système (disque dur, imprimante, base de données...). Par contre, une applet possède un accès restreint aux

ressources du système et ne peut être exécutée que par un navigateur Web compatible Java. Elle doit également être incorporée à une page Web au moyen de balises HTML. Aucun interpréteur Java séparé n'est nécessaire car le navigateur Web en comprend déjà un.

Une technique très simple pour accéder à une base de données à partir d'une application Java est le pont JDBC-ODBC ("Java DataBase Connectivity - Open Data Base Connectivity"). D'après [LEMAY-CADENHEAD 00], ODBC est une "interface commune créée par Microsoft pour accéder aux bases de données SQL et générée par l'Administrateur de source de données ODBC sur un ordinateur fonctionnant sous Windows." Il existe des pilotes qui permettent de faire le lien entre la technologie JDBC et la technologie ODBC. L'Administrateur de source de données ODBC est très simple à utiliser. Ses fonctions sont :

- ajout de pilotes ODBC ;
- configuration de ces pilotes de manière à ce qu'ils fonctionnent avec des fichiers de bases de données précis ;
- conception d'un journal des opérations SQL effectuées.

Le pont JDBC-ODBC permet d'utiliser des pilotes JDBC comme des pilotes ODBC en convertissant les appels de méthode JDBC en appels de fonction ODBC. Pour utiliser ce pont, il faut disposer :

- du pilote du pont JDBC-ODBC (fourni avec Java) ;
- d'un pilote ODBC ;
- d'une source de données ODBC ayant été associée au pilote au moyen de l'Administrateur[7] de source de données ODBC.

Pour pouvoir utiliser une base de données dans une application Java, l'étape suivante consiste à associer cette base de données à la source de données ODBC. Cette étape s'effectue également grâce à l'Administrateur.

---

[7] Il existe d'autres logiciels capables d'associer une source de données ODBC au pilote. Par exemple, cette source de données ODBC peut parfois être configurée au sein même du système de gestion de bases de données.

Au niveau de l'application Java, la connexion à la base de données, ainsi que l'exécution des requêtes et le traitement des résultats s'opèrent selon les étapes suivantes :

- Chargement du pilote de pont JDBC-ODBC ;
- Etablissement de la connexion avec la base de données ;
- Création d'un objet "Statement" qui permet l'exécution d'une requête SQL ;
- Création et exécution d'une requête SQL ;
- Récupération des résultats de la requête ;
- Traitement de ces résultats ;
- Fermeture de la connexion avec la base de données.

Il est certain qu'une application Java ne doit pas obligatoirement utiliser le pont JDBC-ODBC. Elle peut également accéder à une base de données à l'aide de pilotes JDBC de la même manière qu'une applet, comme décrit dans le point suivant. Ces pilotes doivent, dans le cas présent, avoir été implémentés entièrement en Java ou bien à l'aide de code natif.

☙ *Le cas des applets Java : les pilotes JDBC*

Rappelons la définition d'une applet : "Une applet est un programme Java qui s'exécute sur le World Wide Web" [LEMAY-CADENHEAD 00]. Pour fonctionner, celle-ci doit être incorporée à une page Web à l'aide de balises HTML. Quand un utilisateur équipé d'un navigateur compatible Java charge une page Web qui contient une applet, le navigateur télécharge l'appel depuis un serveur Web et l'exécute sur l'ordinateur de l'utilisateur. Le navigateur comprend un interpréteur Java. Par conséquent, le fait qu'aucun interpréteur Java séparé ne soit nécessaire constitue un gros avantage de l'applet. Le cycle de vie d'une applet est décrit dans [ACREMANN-DUPIN-MOUJEARD 99] :

- L'utilisateur demande à charger une page HTML à partir d'un serveur Web ;
- Le code de la page est téléchargé et interprété par le navigateur ;
- Le navigateur trouve la balise APPLET dans la page ;
- Le navigateur télécharge le fichier Java compilé indiqué par la balise ;
- Le navigateur exécute l'applet avec son interpréteur intégré ;

- L'utilisateur visualise l'applet au sein de la page HTML dans le navigateur.

Pour des raisons de sécurité, une applet ne peut pas accéder au système de fichiers du poste d'exécution si elle a été chargée à partir d'un serveur distant [ACREMANN-DUPIN-MOUJEARD 99]. En réalité, si une applet a été chargée à partir d'un serveur, elle ne peut accéder qu'aux classes, ressources et données provenant de ce même serveur. Elle ne peut pas accéder à d'autres serveurs ni au poste d'exécution. Les opérations suivantes lui sont dès lors interdites :

- Lire ou écrire des fichiers sur le système de fichiers de l'utilisateur ;
- Communiquer avec un site Internet autre que celui qui diffuse la page Web dont fait partie l'applet ;
- Faire fonctionner des programmes sur le système de l'utilisateur ;
- Charger des programmes stockés sur le système de l'utilisateur, tels que des programmes exécutables et des bibliothèques partagées.

D'une manière générale, ces normes de sécurité veillent à ce que les applets fonctionnent dans un contexte où la sécurité de l'ordinateur qui les accueille n'est jamais menacée.

De plus, le pilote du pont JDBC-ODBC utilise du code natif et non Java. Or, le code natif ne peut être soumis aux normes restrictives de sécurité imposées par Java. Il n'existe, par conséquent, aucun moyen de garantir sa fiabilité.

En conclusion, le modèle de sécurité auquel sont soumises les applets ne permet pas d'utiliser le pont JDBC-ODBC. C'est pourquoi seuls les pilotes JDBC complètement implémentés en Java peuvent être utilisés dans des applets. Leur avantage est qu'ils ne nécessitent pas d'être configurés sur l'ordinateur client.

Les programmes utilisant des pilotes JDBC sont conçus sensiblement de la même manière que ceux qui reposent sur le pont JDBC-ODBC : les pilotes JDBC sont soit inclus dans des produits commerciaux en vente, soit à télécharger sous forme de versions d'évaluation.

Tout comme dans le cas du pont JDBC-ODBC, il faut configurer une source de données pour JDBC en :

- créant une base de données ;
- associant cette base de données au pilote JDBC ;
- identifiant la source de données par le format de la base de données, le serveur de bases de données, le nom d'utilisateur et le mot de passe.

Pour que le programme fonctionne correctement, il faut d'abord lancer le pilote JDBC. Ensuite, la connexion à la base de données est réalisée de la même manière qu'avec le pont JDBC-ODBC. Une différence existe cependant. Dans le cas d'un pilote JDBC, l'instruction qui demande la connexion doit comprendre une référence vers ce pilote. Cette référence doit indiquer le nom de l'ordinateur ainsi que le numéro de port par défaut que ce pilote utilise. Cette instruction doit aussi contenir l'adresse de la base de données à laquelle il faut se connecter. Les informations de configuration spécifiques au pilote sont fournies par le fabricant de ce pilote.

1.3. Conclusion

Ce stage au Devon County Council fut pour nous une expérience très enrichissante.

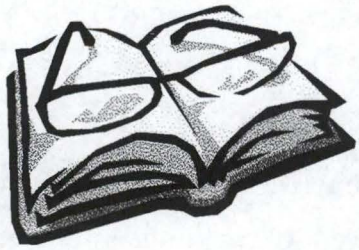
D'une part, le travail effectué dans le cadre de ce stage nous a permis d'élargir nos connaissances en matière de programmation en Java et d'approche "Orienté Objet". En effet, en tant que néophyte, il nous semblait essentiel de parfaire notre formation théorique par une mise en pratique continue et graduellement complexifiée. Nous avons ainsi appris à utiliser des techniques jusqu'alors inconnues. Nos recherches sur quelques sites Internet consacrés à ce langage de programmation ont porté leurs fruits et ont pallié l'absence de programmeurs Java susceptibles de nous procurer aide et conseils.

D'autre part, considérant qu'une bonne maîtrise de la langue anglaise représente un atout majeur pour notre vie professionnelle future, ce stage a également été l'occasion d'améliorer nos compétences linguistiques.

Enfin, d'un point de vue plus personnel, ce stage en Grande-Bretagne nous a apporté beaucoup plus que nous ne l'aurions imaginé. Nous avons découvert un nouveau mode de vie, une culture différente et, avant tout, un peuple chaleureux. Ce séjour de trois mois et demi à Exeter nous a permis de rencontrer nombre de personnes intéressantes et nous a appris à être plus sociable et plus tolérante. En effet, nous avons retiré d'efficaces leçons de savoir-vivre et d'humilité des us et coutumes des Anglais que nous avons côtoyés. Nous avons aussi gagné de la confiance et de l'assurance en voyageant seule et en vivant loin des nôtres.

2

Définition et contenu d'un glossaire



Afin de mieux percevoir tout l'intérêt que comporte la partie glossaire de chaque ouvrage littéraire ou scientifique, ainsi que son utilité, nous avons réalisé, à la bibliothèque d'Exeter, quelques recherches étymologiques et sémantiques portant sur le mot "glossaire". Ce deuxième chapitre en expose les résultats, suivis des constatations que nous en avons faites. Par la suite, nous avons examiné deux cents glossaires sur Internet en vue d'en dénicher des idées intéressantes à appliquer à notre propre glossaire. Ce chapitre se termine par un recensement et quelques critiques de ces idées. Enfin, nous présentons les idées que nous avons adoptées pour notre glossaire.

2.1. Etymologie du mot "glossaire"

Avant toute chose, il nous a paru intéressant de commencer les recherches par une petite étude étymologique du mot "glossaire", à partir du vieil ouvrage [SKEAT 10]. En réalité, la bibliothèque étant située dans un pays anglophone, nos recherches ont été principalement axées sur le mot "glossaire" dans sa traduction anglaise, c'est-à-dire : "glossary".

> **glossary**, a collection of glosses or words explained. (L. – Gk.) In Kersey's Dict. ed. 1715. Spelt *glosarye*, Caxton, Golden Legend, St. Clement, § I.-L. *glōssārium*, a

glossary ; formed with suffix *-āri-um* from L. *glōss-a*, a hard word needing explanation (above). Der. *glossari-al, glossar-ist*.

Comme nous pouvons le constater, un glossaire était, au début du dix-huitième siècle, une collection de gloses ou de mots compliqués expliqués. Pour mieux comprendre cette définition, nous trouvons ci-dessous l'étymologie du mot anglais "gloss", qui signifie "glose".

gloss, a commentary, explanation. (L. – Gk.) ME. *glose* (with one *s*), in early use ; P. Plowman, C. XX. 15. [But the verb *glosen*, to gloss or gloze, was much more common than the sb. ; see Chaucer, C.T. 7374, 7375 (D 1792) ; P. Plowman, B. VII. 303.] This ME. *glose* is from the OF. *glose*, "*a glosse* ;" Cot. But the L. form *glosse* (with double *s*) was substituted for the F. form in the 16th century ; as e.g. in Udal on S. Matt. XXIII. I8. – L. *glōssa*, a difficult word requiring explanation . – Gk. *Γλώσσα*, the tongue ; also, a tongue, language, a word needing explanation. Der. *gloss*, verb ; *gloze*, q.v. ; *gloss-ar-y*, q.v. *; glosso-graphy, glosso-logy ; glottis*, q.v.

Une glose est donc un commentaire ou une explication d'un mot difficile. Uniquement à partir de ces deux étymologies, nous pouvons déjà constater que le mot glossaire, qui vient du latin et du grec, signifie : "Collection d'explications de mots difficiles".

2.2. Définitions du mot "glossaire"

2.2.1. Définitions anglaises

- **Selon [ALLEN 94]**

glossary *noun* (pl. glossaries) a list of glosses, often at the end of a book.

gloss –*noun* a short explanation of a difficult word, phrase, etc. in a text, eg in the margin. –*verb* to provide a gloss of (a word, etc...), or add glosses to (a text). [from Latin *glossa*, word requiring explanation]

Cette définition comporte un détail intéressant. En effet, une glose, c'est-à-dire l'explication d'un mot compliqué, peut se trouver **dans la marge d'un texte**. Ce détail a fait naître une idée nouvelle : afficher dans un "ToolTip" une définition concise de chaque mot d'un texte figurant dans le glossaire.

lexicon *noun* **1** a dictionary. **2** the vocabulary of an individual person, branch of knowledge, or language. [from Greek lexikon, from lexis, word]

En parcourant cette définition, nous pouvons remarquer que le type de glossaire à créer pour le système MAI se rapproche assez de la notion de lexique telle que définie ici : "**vocabulaire d'une connaissance**", la génétique en l'occurrence.

- **Selon [HAWKINS-ALLEN 91]**

glossary *n.* (pl. **-ies**) **1** (also **gloss**) an alphabetical list of terms or words found in or relating to a specific subject or text, esp. dialect, with explanations ; a brief dictionary. **2** a collection of glosses. ♦ **glossarial** adj. **glossarist** n. [L. *glossarium* f. *glossa* gloss]

gloss *n. & v. –n.* **1 a** an explanatory word or phrase inserted between the lines or in the margin of a text. **b** a comment, explanation, interpretation, or paraphrase. **2** a misrepresentation of another's words. **3 a** a glossary. **b** an interlinean translation or annotation. *–v* **1** *tr.* **a** add a gloss or glosses to (a text, word, etc.). **b** read a different sense into ; explain away. **2** *intr.* (often foll. by *on*) make (esp. unfavourable) comments. **3** *intr.* write or introduce glosses. ♦ **glosser** n. [alt. of gloze after med. L. *glossa*]

lexicon *n.* **1** a dictionary, esp. of Greek, Hebrew, Syriac, or Arabic. Until the 19th c. dictionaries of these languages were usually in Latin and entitled *lexicon* rather than *dictionarius*. **2** the vocabulary of a person, language, branch of knowledge, etc. [mod. L f. Gk *lexikon* (*biblion* book), neut. of *lexikos* f. *lexis* word f. *legō* speak]

A l'instar des définitions précédentes, un glossaire est défini, dans [HAWKINS-ALLEN 91], comme une "liste **alphabétique** de termes concernant un sujet ou un texte spécifique, accompagnés d'explications". L'adjectif "alphabétique" ("alphabetic") est un élément nouveau. Il semble néanmoins logique que les termes

d'un glossaire soient classés selon un ordre alphabétique, de manière à en faciliter l'utilisation.

Constatons également que chacune des définitions du mot "glossaire" analysées jusqu'à présent utilise le terme "**explication**" ("explanation") et non le terme "**définition**" ("definition"). Afin de mieux cerner les disparités existant entre ces deux termes, voyons comment [ROBERT 95] les définit :

définition n. f. – XIIe ; de *définir* [...] Formule qui donne le sens d'une unité du lexique (mot, expression) et lui est à peu près synonyme. *Les définitions et les exemples d'un dictionnaire. Définitions et gloses, et traductions. Une définition exacte, approximative.* [...]

explication n. f. – 1322 ; lat. *explicatio* ♦ Action d'expliquer, son résultat. **1.** Développement destiné à faire comprendre qqch. ⇒ **commentaire, éclaircissement**. [...] Explications jointes à un texte (⇒ **annotation, glose, note, remarque, scolie**), à une carte (⇒ **légende**). [...]

A partir de ces deux définitions, force est de constater qu'une explication peut prendre diverses formes (commentaire, annotation, légende...), tandis qu'une définition est une formule assez courte et figée, obéissant à des règles strictes. Par conséquent, le glossaire permet de définir des termes dans le contexte de l'ouvrage auquel il se rapporte et non dans l'absolu à l'instar du dictionnaire.

- **Selon [CAMBRIDGE 95]**

glossary n [C] an alphabetical list, with meanings, of the words or phrases in a text that are difficult to understand · *The book would have been more useful if a glossary of technical terms and abbreviations had been included.*

gloss n, v (to provide) an explanation for a word or phrase. *There is a special edition of the book for schools, in which the older and more rare words have been glossed.* [T] • *Expressions that are difficult to understand are explained in the glosses at the bottom of the page* [C] •

lexicon n [C] specialized (a list of) all the words used in a particular language or subject, or a dictionary.

Dans [CAMBRIDGE 95], nous retrouvons la notion de liste **alphabétique** de termes, mais ici, ces derniers sont clairement spécifiés comme "**difficiles à comprendre**" ("difficult to understand"). Une fois de plus, par cette caractéristique, le glossaire se distingue du dictionnaire par le fait qu'il ne répertorie qu'un sous-ensemble de termes, les plus compliqués. En effet, le dictionnaire, même s'il est restreint à un domaine particulier (par exemple, un dictionnaire médical), donne la définition de tous les termes appartenant à ce domaine.

Par ailleurs, dans [CAMBRIDGE 95], le terme "signification" ("meaning") remplace le vocable "explication" observé chez les autres auteurs. Portons notre attention sur la définition que nous donnent [LAROUSSE 87] et [ROBERT 95] de la "signification", selon l'acception qui nous concerne. :

signification n. f. Sens et valeur d'un mot.

signification n. f. – déb. XIIe ; lat. *significatio* **1.** [...] ◊ Sens (d'un signe, d'un ensemble de signes et SPECIALT d'un mot). ⇒ **contenu**. *Les diverses significations d'un mot.* ⇒ **polysémie**.◊ LING. Rapport réciproque qui unit le signifiant et le signifié. [...]

Ces définitions ne nous éclairent pas grandement sur les formes que peut prendre une signification. Voyons maintenant la définition du "sens", terme le plus proche de "signification" selon les deux définitions précédentes (d'après [LAROUSSE 87] et [ROBERT 95]) :

sens n. m. (lat. *sensus*). [...] *Ling.* et *Log.* Ensemble des représentations que suggère un mot, un énoncé.

sens n. m. – 1080 ; lat. *sensus*, de *sentire* → sentir. [...] **III. 1.** Idée ou ensemble d'idées intelligible que représente un signe ou un ensemble de signes. ⇒ **signification**. *Chercher le sens d'un mot, d'une expression, d'une phrase. Avoir du sens, un sens.* (⇒ **signifiance**).

Au vu de ces définitions, nous pouvons davantage associer le terme "signification" au terme "explication" qu'au mot "définition". En effet, les vocables "représentation", "idée" et "ensemble d'idées" représentent des concepts assez vagues pouvant très bien regrouper les notions d'éclaircissement, remarque, légende... De plus, cet ensemble d'idées doit être "intelligible", ce qui dénonce bien la volonté d'expliquer des termes difficiles à comprendre.

- **Selon [BARNHART 92]**

glossary *n., pl.* **–ries**. **1** a list of special, technical, or hard words, usually in alphabetical order, with explanations or comments ; collection of glosses : *a glossary to Shakespeare's plays. Some textbooks have glossaries at the end.* SYN : gloss, key. **2** a vocabulary or dictionary of limited scope : *a glossary of Scottish words, a glossary of the mining and mineral industry.* [< Latin *glōssārium* < *glōssa* ; see etym. under gloss].

gloss *n., v. –n.* **1** a word inserted between the lines or in the margin of a text to explain or give the equivalent of a foreign or difficult word. **2** *Figurative* **a** a comment ; explanation ; interpretation : *in the margins of library books earnest freshmen inscribed such helpful glosses as "Description of nature", "Irony", and "How true !"* (New Yorker). **b** an artfully misleading or false interpretation. **3** = glossary. **4** an interpretation or definition of a word or phrase given in a glossary or dictionary. **5** a translation inserted between the lines of a text printed in a foreign language. **6** a series of verbal interpretations of a text.
–v.t. **1** to insert glosses or comments on ; ex-plain. SYN : interpret, annotate. **2** to explain away ; read a different sense into ; misinterpret.
-v.i. to make glosses, comment. [< Latin *glōssa* < Greek *glôssa* obsolete or foreign word that needs explaining ; (literally) tongue. See etym. of doublet gloze].

lexicon *n.* **1** a dictionary, especially of Greek, Latin, or Hebrew. SYN : Wordbook **2** the vocabulary of a language or of a certain subject, group, or activity : *In the lexicon of youth...there is no such word as –fail !* (Edward Bulwer-Lytton). **3** *Linguistics.* the total stock of morphemes in a given language. [< Greek *lexikón* (*biblión*) wordbook, neuter of *lexikós* of words < *léxis* word < *leg-*, stem of *légein* say]

[BARNHART 92] confirme nos précédentes déductions. De fait, un glossaire est une liste de mots **complexes, classés par ordre alphabétique**. Cependant, cette définition apporte un élément nouveau : les termes d'un glossaire peuvent également être, en sus de complexes, **spéciaux** ou **techniques**. Nous ne jugerons pas de la pertinence de cette distinction, qui semble ne pas avoir été mentionnée par les autres auteurs. Ces derniers ont probablement regroupé, de manière sémantique, ces termes spéciaux et techniques dans l'ensemble des termes complexes.

Par ailleurs, un glossaire, selon ce dictionnaire, peut être le **vocabulaire** d'un domaine limité. Les exemples montrent qu'un glossaire peut être le vocabulaire d'une langue ou d'un dialecte, mais aussi le vocabulaire d'un domaine d'activités comme, par exemple, l'industrie minière. Dans notre cas, le glossaire MAI revêtira aussi une fonction de vocabulaire. Il faut savoir qu'en réalité, le glossaire MAI se composera de plusieurs glossaires, chacun étant dédié à un seul cours. On peut donc considérer un cours comme un domaine d'activités restreint (par exemple, la biologie).

- **Selon [SIMPSON-WEINER 89]**

glossary Also **glosarye**. [ad. L. *glossārium*, f. *glōssa* gloss sb. : see –ary. Cf. F. *glossaire*.] A collection of glosses ; a list with explanations of abstruse, antiquated, dialectal, or technical terms ; a partial dictionary. [...]

gloss *n.* *******

v. [f. gloss sb.]

1. **a.** *trans.* To insert glosses or comments on ; to comment upon, explain, interpret ; = gloze. [...]

   b. *intr.* To introduce a gloss, comment, or explanation upon a word or passage in a text. Also in wider sense, to make comments or remarks (esp. unfavourable ones) *upon* a person's words or actions. Const. *on, upon,*†*at*. (Cf. gloze).[...]

2. *trans.* To veil with glosses ; to explain away ; to read a different sense into. Also with *away, over* [...].

lexicon [? mod.L., a. Gr. λεξικόν (sc. βιβλίον), neut. sing. of λεξικός of or for words, f. λέξι–ς diction, word, phrase, f. λεγ- to speak.]

1. **a.** A word-book or dictionary ; chiefly applied to a dictionary of Greek, Hebrew, Syriac, or Arabic. The restricted use is due to the fact that until recently dictionaries of these particular languages were usually in Latin, and in mod.L. lexicon, not dictionarius, has been the word generally used. [...]

b. *fig.* (*a*) The vocabulary proper to some department of knowledge or sphere of activity ; the vocabulary or word-stock of a region, a particular speaker, etc. (*b*) A list of words or names. [...]

2. *Linguistics.* The complete set of meaningful units in a language ; the words, etc., as in a dictionary, but without the definitions. (Opp. grammar *sb.*) [...]

3. (With capital initial) The proprietary name of a game played with cards marked with the letters of the alphabet. [...]

Dans le dictionnaire [SIMPSON-WEINER 89], les adjectifs qualifiant les termes expliqués dans un glossaire sont particulièrement imagés et peu usités : "abstrus" ("abstruse"), qui signifie "difficile à comprendre" (syn : "abscons", "abstrait", "obscur") - "désuet" ("antiquated") (syn : "vieilli", "suranné", "d'autrefois") - "dialectal" ("dialectal"), qui vient d'un dialecte - "technique" ("technical"). Néanmoins, la plupart de ces adjectifs ne sont que des nuances de ceux déjà rencontrés dans les définitions analysées ci-avant. Pourtant, "désuet" et "dialectal" apparaissent comme des caractéristiques nouvelles des termes d'un glossaire. Nous ne nous attarderons pas sur ces adjectifs car ils ne concernent pas (ou très peu) le glossaire MAI.

- **Selon [NORTON-ESPOSITO 95]**

[...] "**Dictionary**" is used to describe a wide variety of reference works. Basically, a dictionary lists a set of words with information about them. The list may attempt to be a complete inventory of a language or may be only a small segment of it. A short list, sometimes at the back of a book is often called a **glossary**. When a word list is an index to a limited body of writing, with references to each passage, it is called a **concordance**. [...] A word list that consists of geographic names only is called a **gazetteer**. The word **lexicon** designates a wordbook, but it also has a special abstract meaning among linguists, referring to the body of separable structural units of which the language is made up. [...]

dictionary, reference book that lists words in order and gives their meanings. In dictionaries of Western languages, the words are given in alphabetic order. In addition to its basic function of defining words, a dictionary may provide information about their pronunciation, grammatical forms and functions, etymologies, syntactic peculiarities, variant spellings, conventional abbreviations, and synonyms and antonyms. A dictionary may also provide quotations illustrating a word's use. [...]

Malgré la distinction stricte établie précédemment entre un glossaire et un dictionnaire, les deux définitions d'un dictionnaire données par [NORTON-ESPOSITO 95] révèlent que le glossaire MAI présentera également des caractéristiques propres au dictionnaire. En effet, il comportera une liste des mots utilisés dans un domaine spécialisé et leurs explications, mais aussi des traductions, synonymes, antonymes, illustrations... Le contenu du glossaire à créer est beaucoup plus complet et varié que celui d'un glossaire ordinaire. Par conséquent, nous pouvons affirmer que le glossaire MAI sera le résultat d'une fusion entre un glossaire et un dictionnaire.

2.2.2. Définitions françaises

- **Selon [ROBERT 95]**

glossaire [glɔsɛr]. n.m. (1664 ; *glosaire*, XVIe ; bas lat. *glossarium*). Dictionnaire qui donne l'explication de mots anciens, spéciaux ou mal connus. *Le glossaire du bas latin de Du Cange*. ◊ Lexique d'une langue vivante, *spécialt*. d'un dialecte ou patois. ◊ Lexique d'un domaine spécialisé. *Glossaire de génétique*.

lexique [lɛksik]. n.m. (1721 ; *lexicon*, 1563 ; gr. *lexikon*, de *lexis* "mot"). ♦ **1°** *Vx*. Dictionnaire. ◊ *Mod*. Dictionnaire succinct (d'une science ou d'une technique, d'un domaine spécialisé). V. **Glossaire** – Dictionnaire bilingue abrégé. V. **Vocabulaire**. – Recueil des mots employés par un auteur, dans une œuvre littéraire. V. **Index**. *Un lexique de Cicéron, de La Bruyère*. ♦ **2°** (1861). *Ling*. L'ensemble indéterminé des éléments signifiants stables (mots, locutions...) d'une langue, considéré abstraitement

comme une des composantes formant le code de cette langue. V. aussi **Vocabulaire**. *Etude du lexique*. V. **Lexicographie, lexicologie**. ♦ **3°** Ensemble des mots employés par qqun. *Un lexique de quatre mille mots. La richesse du lexique de Proust.* (V. idiolecte).

Le glossaire MAI englobe la première et la troisième acceptions du mot "glossaire" proposées par [ROBERT 95]. De fait, il s'agit d'un lexique d'un domaine spécialisé - la génétique en l'occurrence -, mais son utilité principale est d'expliquer à son utilisateur la signification de mots mal connus.

- **Selon [LAROUSSE 73]**

glossaire [glɔsɛr]. n.m. (lat. impér. *glos(s)arium*, dictionnaire où on explique les termes rares ou vieillis, de *glos(s)a* [v. glose] ; v. 1585, Cholières, écrit *glosaire*, au sens de "recueil de gloses" ; écrit *glossaire*, au sens 1, 1680, Richelet ; sens 2, 1835, Acad. ["petit lexique d'un auteur à la fin d'une édition classique", fin du XIXe s.]). **1.** Dictionnaire expliquant les mots vieillis ou obscurs d'une langue : *Le glossaire de Ch. Du Cange. Fouillons les chartiers, refouillons les glossaires* (Hugo). *Gautier [...] estimait indigne de vivre tout poète [...] qui ne prend pas plaisir à lire les lexiques et les glossaires* (France). || **2.** *Spécialem.* Lexique d'un dialecte ou d'un patois, ou d'un langage spécial. || Petit lexique d'un auteur à la fin d'une édition classique.

glose [gloz] n.f. (bas lat. *glosa*, var. de *glossa*, mot rare qui a besoin d'une explication [en lat. class. le plur. *glossae* signifiait "glossaire"], gr. *glôssa*, langue [organe], langage ; XIIe s., Everat, au sens de "interprétation symbolique d'un récit biblique" ; sens 1-2, début du XIIIe s., *Barlaham* ; sens 3, v. 1220, Guiot de Dijon ; sens 4, 1680, Richelet). **1.** Annotation, généralement très concise, ajoutée à un texte pour en éclairer les mots ou passages obscurs : *Une glose marginale, interlinéaire. Les gloses des Pères de l'Église sur l'Écriture.* || **2.** Note ou commentaire explicatif ou critique d'un texte : *Ce fut un temps de théories, de curiosités, de gloses et d'explications passionnées* (Valéry). *Quiconque en France, publie des gloses sur une de ses oeuvres s'expose de façon automatique aux ricanements* (Montherlant). || **3.** *Vx.* Commentaire inutile ou malveillant : *Un texte où chacun fait sa glose* (Boileau). *Redouter les gloses du public.* || **4.** *Spécialem.* et *vx.* Parodie rimée où chaque stance est le commentaire burlesque d'un

vers de la poésie parodiée : *La glose de Sarrasin sur le sonnet de Job.* • **SYN.** : **1** *note, scolie* ; **2** *explication, interprétation.*

lexique [lɛksik] n.m. (gr. *lexikon*, lexique, neutre substantivé de l'adj. *lexikos*, qui concerne les mots [v. lexic(o)-] ; milieu du XVIe s., Ronsard, écrit *lexicon* [*lexique*, XVIIIe s.], au sens 3 ; sens 1, 1580, Montaigne, écrit *lexicon* [*lexique*, 1721, Trévoux] ; sens 2, 1893, *Dict. général* ; sens 4, 1867, Littré ; sens 5, 1861, Baudelaire ["ensemble des éléments enregistrés dans un dictionnaire...", milieu du XXe s.]). **1.** *Vx.* Dictionnaire bilingue, grec ou latin. || **2.** Dictionnaire bilingue abrégé : *Il s'était débrouillé tout seul en s'aidant d'un méchant lexique* (Dorgelès). || **3.** Liste des mots d'un ouvrage qui s'écartent de l'usage commun, avec l'explication de ces mots : *Un roman policier suivi d'un lexique des mots d'argot. Beaucoup ont illustré ses fables de notes, gloses et lexiques* (France). || **4.** Recueil des mots employés par un auteur : *Un lexique de Corneille, de Racine.* || **5.** Ensemble des mots d'une langue : *Les deux tiers du lexique anglais sont d'origine romane.* || *Spécialem.* Ensemble des éléments enregistrés dans un dictionnaire (liste ouverte), par opposition aux éléments enregistrés dans une grammaire (liste fermée). [v. art. spécial.] • **SYN.** : **3** *glossaire* ; **4** *index* ; **5** *vocabulaire.*

dictionnaire [diksjɔnɛr] n. m. (lat. médiév. *dictionarium*, dér. de *dictio*, mot [v. diction] ; 1539, R. Estienne, au sens 1 ; sens 2-3, fin du XVIe s., sens 4, 1762, J.-J. Rousseau). **1.** Recueil des mots et des expressions d'une langue, ou d'une catégorie d'entre eux, généralement rangés par ordre alphabétique (ou selon un ordre différent, mais permettant de les retrouver aisément) et suivis de leur définition dans cette langue, ou de leur traduction dans une langue étrangère : [...] *On y cherche le sens des mots, la génération des mots, l'étymologie des mots ; enfin on en extrait tous les éléments qui composent une phrase et un récit* [...] (Baudelaire). [...] || *Dictionnaire de la langue*, ouvrage qui donne et définit les mots de la langue commune. || *Dictionnaire bilingue*, ouvrage qui donne la traduction des mots d'une langue dans une autre, avec des locutions et des expressions où ils sont employés : *Dictionnaire latin-français* [...]. || **2.** Ouvrage analogue relatif à un aspect particulier d'une langue : *Dictionnaire des synonymes. Dictionnaire del'argot. Dictionnaire de la langue philosophique.* || *Dictionnaire étymologique*, celui qui donne l'origine des mots d'une langue, généralement avec la date de l'apparition des sens. || **3.** Ouvrage relatif à une matière,

une science, un art et dont les articles sont disposés par ordre alphabétique : *Dictionnaire historique, géographique. Dictionnaire des oeuvres.* || Spécialem. *Dictionnaire encyclopédique*, ouvrage qui, outre les définitions de mots, contient des développements d'ordre scientifique ou technique, historique, géographique, littéraire, etc., ainsi que des notices biographiques , et donne une somme des connaissances à une époque. || **4.** *Vx.* L'ensemble du vocabulaire employé par quelqu'un : *Le dictionnaire de Bossuet.*

Comme nous pouvons le constater dans [LAROUSSE 73], le glossaire est défini comme un dictionnaire limité aux définitions de termes vieillis, rares ou obscurs. Un dictionnaire est, selon cet ouvrage, un "recueil des mots et des expressions d'une langue, ou d'une catégorie d'entre eux, généralement rangés par ordre alphabétique [...] et suivis de leur définition dans cette langue, ou de leur traduction dans une langue étrangère". Cette définition, malgré sa longueur, est trop restrictive par rapport aux acceptions proposées par les autres auteurs. Elle ne tient pas compte des termes spéciaux et techniques et se limite aux seules définitions ou, à la rigueur, aux traductions. Nous ne pouvons, dès lors, rien en retirer.

- **Selon [LAROUSSE 89]**

glossaire n.m. (bas lat. *glossarium*, de *glosa*, glose). **1.** Philol. Recueil de gloses. – **2.** Liste alphabétique placée à la fin d'un ouvrage et donnant les mots du vocabulaire spécialisé qui y est utilisé. – **3.** Nom donné à certains dictionnaires alphabétiques portant sur un domaine spécialisé, à certains dictionnaires bilingues, en particulier dans le domaine dialectal.

glose n.f. (bas lat. *glosa* ; du gr. *glôssa*, langue, language). **1.** Explication de quelques mots obscurs d'une langue par d'autres mots plus intelligibles. –**2.** Commentaire, notes servant à l'intelligence d'un texte.

–Littér. Parodie d'une pièce de poésie, où chaque vers de la pièce donne lieu à une strophe qui en est le commentaire plaisant.
–Philol. Annotation très concise que contiennent certains manuscrits entre les lignes ou en marge, et visant à expliquer au lecteur un mot ou un passage jugé obscur. Elle est

parfois introduite dans le texte même . (Coleridge a publié simultanément son poème *Le Vieux Marin* [1798] et sa glose sur ce poème.) [v. part. *encycl.*]

♦ **gloses** n.f.pl. [...]

–ENCYCL. Dr. anc. et Hist. Certains manuscrits de la loi salique sont accompagnés d'annotations en dialecte franc, connues sous le nom de *gloses malbergiques de mallus*, tribunal). Les gloses ont été en usage dans les anciennes écoles de droit du Moyen Age où fut étudiée l'œuvre de Justinien (→ *école de* BOLOGNE). Elles n'étaient, au début, que de courtes explications sur des mots difficiles, placées entre les lignes du texte (gloses interlinéaires) ; plus tard, ce furent des explications plus étendues et en marge (gloses marginales). Au XIIIe s., les gloses envahirent tellement les textes que ceux-ci disparurent presque sous leur accumulation.

lexique n.m. (gr. *lexikon*, de *lexis*, parole, mot). **1.** Dictionnaire spécialisé et généralement succinct concernant une science, une technique, un domaine quelconque de la connaissance. –**2.** Dictionnaire bilingue succinct réduit à la mise en parallèle des unités lexicales des deux langues confrontées. –**3.** Ensemble ordonné des mots employés dans une œuvre littéraire, par un écrivain. –**4.** Nom parfois donné à un glossaire placé en fin d'ouvrage. –**5.** Ling. Ensemble des unités significatives formant la langue d'une communauté et considéré abstraitement comme l'un des éléments constituant le code de cette langue. [...]

La première acception du terme "glossaire" présentée par cet auteur : "recueil de gloses" semble être la seule à correspondre au glossaire MAI. Effectivement, une glose est, d'après [LAROUSSE 89], l'explication d'un terme obscur par d'autres mots plus intelligibles ou bien un commentaire, une annotation permettant la compréhension d'un texte. Comme nous l'avons vu auparavant, le glossaire est parfois aussi appelé lexique. Tous ces aspects ont déjà été analysés précédemment. Nous n'y reviendrons donc pas.

2.2.3. Conclusions

Après avoir parcouru ces nombreuses définitions, chacune d'elles apportant un complément d'information aux autres définitions, nous pouvons en tirer quelques conclusions intéressantes.

Tout d'abord, un glossaire est une collection de gloses. Une glose est une explication d'un mot difficile, spécial, ancien, dialectal ou technique. Un glossaire est, par conséquent, une sorte de dictionnaire restreint à un nombre limité de mots reliés sémantiquement. Il existe trois types de glossaires distincts selon le rôle qu'ils jouent :

- Le "dictionnaire bilingue" : glossaire donnant la traduction française de mots d'une autre langue (morte ou vivante) ou d'un dialecte.
- Le "lexique" : glossaire répertoriant les mots utilisés par une personne. Il s'agit souvent du vocabulaire employé par un auteur dans ses ouvrages.
- Le "vocabulaire" : glossaire expliquant tous les termes propres à un domaine de connaissances précis ou une sphère d'activités.

Le glossaire MAI, malgré sa large interprétation, est plutôt un glossaire du troisième type. En effet, il a pour objectif de permettre à l'étudiant d'un cours de génétique d'obtenir des compléments d'information au sujet d'un terme inconnu ou mal maîtrisé. La génétique est, bien entendu, un domaine de connaissances particulier, sous-ensemble des sciences.

2.3. Recherches de nouvelles idées sur Internet

Nos recherches sur Internet nous ont permis de découvrir de nombreuses idées originales à exploiter. Nous avons visionné quelques centaines de sites incluant des glossaires[8]. Le lecteur trouvera ci-dessous une analyse des concepts notables découverts lors de ces recherches.

[8] Les adresses des sites les plus intéressants, ainsi que des illustrations de leurs glossaires peuvent être consultées dans l'annexe A : "Glossaires électroniques".

Glossaire 1 :

- Les mots de la définition possédant eux aussi une définition propre dans le glossaire présentent un lien hypertexte vers cette définition. Ce lien hypertexte est représenté de manière standard par un mot souligné et d'une couleur différente (bleu en général) de celle utilisée pour les autres mots du texte ;
- Section proposant tous les termes apparentés se trouvant dans le même site Web : "Related items" ;
- Section proposant d'autres sites Web traitant du même sujet : "Internet sites".

Glossaire 2 :

- La définition dans le glossaire est très exhaustive et divisée en différentes sections possédant chacune un titre. Au début de chaque section, une invite offre la possibilité d'imprimer cette section ;
- Un plan d'ensemble permet de visualiser le schéma d'articulation de la définition et de naviguer plus rapidement d'une section à l'autre ;
- De petites photographies et/ou images illustrent la définition. Il est également possible de les visionner à plus grande échelle. Une courte légende apparaît alors en bas de l'image ou de la photographie.

Glossaire 3 :

- Tout comme dans le système MAI, ce site Web s'articule au moyen d'onglets. L'un d'eux conduit également vers le glossaire ;
- Une liste de lettres (A, B, C, D, E, F, GH, ...) permet à l'utilisateur de se déplacer dans le glossaire ;
- Une page différente apparaît pour chaque lettre (ou groupe de lettres) sélectionnée dans la liste de lettres ;
- Chaque page est dessinée selon un même canevas : une colonne à gauche propose une liste de tous les mots commençant par une même lettre et un panneau à droite présente la définition du mot sélectionné ;
- Une petite annotation explique à l'utilisateur comment se servir de ce glossaire : "Il faut cliquer sur l'un des mots dans la colonne de gauche pour afficher sa définition";

- Dès que l'utilisateur clique sur l'un de ces mots, sa définition est affichée dans le panneau de droite.

Glossaire 4 :

- Une longue liste de lettres ou groupes de lettres (Aa- Ad- Al-...) permet de se déplacer dans le glossaire ;
- Quand on choisit une lettre ou un groupe de lettres, tous les mots commençant par ces lettres apparaissent l'un à la suite de l'autre dans une longue liste horizontale, ce qui n'est ni joli, ni aisé à parcourir ;
- Après avoir cliqué sur l'un de ces mots, sa définition se manifeste sur la même page, juste en-dessous de la liste de mots. A nouveau, cette présentation n'est pas des plus ergonomiques ;
- La définition peut contenir des liens hypertextes vers des mots apparentés (synonymes ou mots utilisés dans le même domaine sémantique).

Glossaire 5 :

- Les définitions sont clairement séparées par un double trait.

Glossaire 6 :

- Quand l'utilisateur tape les lettres du mot dont il désire connaître la signification, la liste des mots se déroule pour trouver ce mot ;
- Une barre de défilement permet à l'utilisateur de voir la définition entière, ainsi que de parcourir la liste des mots.

Glossaire 7 :

- L'utilisateur n'a qu'une seule possibilité dans ce glossaire : taper le mot dont il souhaite connaître la définition et ensuite soumettre sa requête ;
- Une liste de mots correspondant à sa requête est affichée. L'utilisateur doit alors choisir le terme qu'il désire ;
- Ensuite, la définition apparaît, ainsi que les mots apparentés (dans un panneau distinct et de couleur différente). Ces derniers, de même que les mots de la

définition qui possèdent une définition dans le glossaire correspondent à des liens hypertextes.

Glossaire 8 :

- La liste des lettres se trouve également à la fin d'un article de manière à alléger la charge de travail de l'utilisateur. En effet, il n'a plus besoin de remonter au début de la page pour atteindre la liste des lettres (en utilisant la barre de défilement) ;

Glossaire 9 :

- L'utilisateur se voit offrir deux possibilités pour obtenir la définition d'un mot inconnu : une liste de lettres ou une option de recherche présentant un champ à remplir.

Glossaire 10 :

- Des images en couleur peuvent être insérées dans la définition d'un mot pour l'illustrer.

Glossaire 11 :

Ce glossaire inclut :

- la prononciation phonétique de certains mots plus compliqués que d'autres ;
- des références internes et des mots-clés avec liens hypertextes ;
- des exemples ;
- des citations ;
- des directives dépendant du sujet du dictionnaire ;
- des informations complémentaires ;
- une présentation très colorée et attrayante.

Glossaire 12 :

- Si, comme dans ce glossaire, nous ajoutons à notre glossaire les traductions d'un mot en d'autres langues, nous pouvons l'appeler "glossaire plurilingue" ;

- A chaque terme technique ou médical du glossaire, est associé le terme familier correspondant (par exemple : méd.abdomen – fam.ventre).

Glossaire 13 :

- Les termes du glossaire sont parfois accompagnés de leur abréviation ;
- Les définitions des termes apparentés sont affichées aux côtés de la définition du mot demandé. Cette présentation est plutôt inadaptée.

Glossaire 14 :

- L'illustration associée à la définition se trouve sur une page séparée à laquelle l'utilisateur peut accéder au moyen d'un hyperlien.

Glossaire 15 :

- La prononciation complète la définition de certains mots du glossaire.

Glossaire 16 :

- Présence et traduction des origines latines ou grecques des mots du glossaire.

Glossaire 17 :

- On doit choisir le mot (ou la lettre) voulu(e) dans une liste déroulante se trouvant dans la partie gauche du canevas. Seule la partie droite du canevas est modifiée pour atteindre la définition du mot ;
- Dans certaines définitions, des comparaisons sont présentées sous forme d'un tableau ;
- Des exemples illustrent la signification du mot.

Glossaire 18 :

- La traduction du mot et de la définition dans trois autres langues est affichée suite à un clic sur le drapeau symbolisant la langue désirée ;

- Un ou plusieurs synonymes de ce mot sont présentés.

Glossaire 19 :

- Les liens hypertextes sont précédés de : "Cf.".

Glossaire 20 :

- Les traductions en latin et en anglais sont proposées à l'utilisateur ;
- L'utilisateur est informé de la date de la première apparition du mot dans la littérature;
- Les illustrations sont de bonne qualité ;

2.4. Idées adoptées par l'équipe MAI

Le stage terminé, le temps était venu de discuter, avec les autres membres de l'équipe MAI, des idées recueillies lors des recherches réalisées à Exeter. Après de mûres réflexions et de longues réunions, nous avons tout d'abord décidé, d'un commun accord, qu'il y aura deux manières d'accéder à la définition d'un terme du glossaire d'un cours : par l'interface du glossaire qui devient disponible lorsqu'on clique sur l'onglet "glossaire" et à partir d'une unité d'apprentissage[9]. Cette deuxième possibilité consistera à associer à chaque terme d'une unité d'apprentissage appartenant au glossaire, un lien hypertexte vers celui-ci. Ce lien sera illustré dans l'unité d'apprentissage par un soulignement et une coloration particulière du terme. De plus, déplacer le curseur sur cet élément, sans "cliquer" dessus, affichera sa définition dans un "ToolTip", petit encart textuel apparaissant à l'écran. En ce qui concerne l'accès direct au glossaire MAI, via l'onglet "glossaire", il a été décidé d'appliquer les idées suivantes, qui constitueront le point de départ de l'élaboration des spécifications : le mode de sélection d'un terme, le canevas que respectera l'interface, l'abréviation du terme, les traductions du terme dans certaines langues, le format de la définition,

[9] Une unité d'apprentissage peut être assimilée à un chapitre d'un cours. Cette notion sera détaillée dans le chapitre 3.

l'ensemble des termes apparentés et opposés, et enfin la liste des unités d'apprentissage référencées[10].

2.4.1. Choix du terme

L'utilisateur du glossaire pourra choisir le terme pour lequel il désire obtenir des explications de deux manières différentes. La première possibilité consistera à choisir le terme parmi tous les termes du glossaire représentés sous forme de liste. Grâce à deux barres de défilement, cette liste sera aisée à parcourir. S'il adopte la seconde alternative, l'utilisateur devra taper le mot dans un champ de saisie, ce qui fera dérouler une liste de termes (à la recherche de ce mot). Dès qu'un terme aura été sélectionné, tous ses attributs seront affichés sur l'interface.

L'idée de la liste de lettres n'a pas été adoptée car elle n'est utile que dans le cas de glossaires imposants. Or, le glossaire MAI se décompose en petits glossaires, chacun appartenant à un cours bien particulier. Il n'est donc pas nécessaire de prévoir une liste de lettres pour faciliter la navigation de l'utilisateur dans le glossaire. De plus, le champ de saisie remplace largement cette fonctionnalité.

2.4.2. Canevas

L'interface sera toujours dessinée selon le même canevas, quel que soit le terme sélectionné : un champ de saisie et une liste déroulante se trouveront sur la partie gauche de l'interface et tous les attributs du terme sélectionné seront affichés dans la partie droite de l'interface, partie qui sera beaucoup plus grande.

2.4.3. Abréviation

Chaque terme du glossaire MAI possédera éventuellement une abréviation ou un acronyme[11]. Par contre, il ne nous a pas semblé pertinent, dans un premier temps,

---

[10] Les idées adoptées seront appliquées tant à l'interface "Apprenant" qu'à l'interface "Professeur", même si parfois l'une des deux interfaces sera plus concernée que l'autre. La distinction entre ces deux interfaces sera explicitée dans les chapitres suivants.

[11] Le professeur qui crée un glossaire est libre de spécifier ou pas une abréviation pour chaque terme du glossaire. Il en va de même pour les autres attributs des termes du glossaire.

de fournir la prononciation des termes, le terme familier associé, l'origine grecque ou latine du mot, ni la date de la première apparition du mot dans la littérature.

2.4.4. Traductions

Des traductions anglaise, néerlandaise, allemande et espagnole seront disponibles pour chaque terme du glossaire. Il suffira de choisir le drapeau apparenté à la langue désirée pour apercevoir la traduction correspondante. Le glossaire MAI sera donc un glossaire plurilingue.

2.4.5. Définition

Chacun des termes du glossaire MAI possédera une définition pouvant être agrémentée d'une illustration ou d'une image animée. La définition, ainsi que l'illustration, seront munies de deux barres de défilement. Ces dernières permettront de visualiser entièrement la définition et l'illustration lorsque celles-ci seront plus grandes que l'espace qui leur a été réservé.

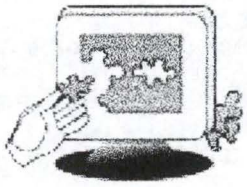
2.4.6. Termes apparentés et termes opposés

Le glossaire MAI mettra à disposition, pour chaque terme qu'il contient, une liste de termes apparentés (synonymes, termes appartenant au même domaine sémantique...) et une liste de termes opposés (antonymes). Ces termes ne feront pas obligatoirement partie des termes du glossaire.

2.4.7. Unités d'apprentissage référencées

Comme nous le verrons dans le chapitre suivant, chaque cours présent dans le système MAI se compose d'unités d'apprentissage. Ces unités d'apprentissage peuvent, bien entendu, traiter de certains termes contenus dans le glossaire. C'est pourquoi nous mettrons à disposition de l'utilisateur une liste d'unités d'apprentissage référencées (maximum quatre) pour chacun des termes du glossaire. Nous permettrons à l'utilisateur du glossaire d'accéder à un aperçu de ces unités d'apprentissage à partir du glossaire.

3
Le projet MAI



Le glossaire MAI, objet principal de ce travail, n'aurait aucune raison d'être sans le système MAI, qui l'accueillera en son sein. De fait, malgré sa complexité et son ampleur, le glossaire ne constituera qu'une des nombreuses fonctionnalités offertes par le logiciel réalisé par l'équipe MAI. Il devra fusionner complètement avec ce système de manière à former un programme homogène. Il semble donc pertinent d'introduire l'analyse et la conception du glossaire MAI par un portrait global de son hôte. Pour réaliser ce portrait, nous nous sommes permise de consulter de nombreux documents appartenant au projet MAI et d'en tirer une synthèse personnelle. Ce troisième chapitre commence par présenter le projet MAI et ses volets biologie et informatique. Ensuite, il décrit les scénarios de l'apprenant et du professeur, ainsi que les technologies et langages de programmation utilisés. Pour finir, il explique l'architecture client-serveur et le dictionnaire de données du système MAI.

3.1. Description du projet MAI

Le projet MAI ("Modules d'Apprentissage Interactif") consiste en la réalisation d'un système auteur qui fonctionne à partir du Web et dont le but premier est d'aider des non-informaticiens à développer des modules de formation. Ce système porte le nom de "système auteur" car il a pour but d'aider à l'élaboration de ces modules. Il permet d'exploiter, d'organiser et de relier différents types de données tels que le texte, l'image et le son de manière à rendre cet ensemble interactif. Le système auteur

fonctionne selon deux niveaux : celui du professeur et celui de l'apprenant. Le professeur utilise le système auteur pour créer des modules de cours interactifs, des questionnaires d'évaluation, ainsi que la carte de navigation suggérant le parcours sémantique que l'élève doit suivre lors de son apprentissage d'un cours. Le professeur peut réutiliser des objets existants (modules de cours, illustrations, simulations...) qu'il va combiner de façon à créer son cours, mais aussi créer personnellement de tout nouveaux modules d'apprentissage qu'il assemblera à sa guise. De son côté, l'apprenant consulte les modules de cours en respectant le parcours sémantique défini par l'auteur du cours. Il peut également, à la fin de chaque cours, évaluer son niveau d'apprentissage au moyen du questionnaire d'évaluation élaboré par l'auteur du cours. De plus, à tout moment, il peut consulter le glossaire pour obtenir des explications sur un terme précis.

Le deuxième objectif du projet est de s'éloigner de l'approche linéaire des livres ou syllabi. En effet, le découpage d'un cours en différents modules, organisés selon un réseau sémantique, transforme l'apprentissage linéaire et statique des manuels en un enseignement personnalisé, en fonction des objectifs et acquis de l'apprenant. En outre, ce parcours individualisé, matérialisé par une carte de navigation, offre à l'apprenant la possibilité de contrôler son avancement dans l'apprentissage d'un cours et d'évaluer son niveau d'apprentissage des concepts déjà vus. L'innovation pédagogique apportée par le système MAI, par rapport aux autres produits de formation multimédias mis sur le marché, repose sur cette structure non linéaire, illustrée par la carte de navigation. De fait, la plupart des logiciels de formation conservent la structure linéaire calquée sur les manuels et suscitent, dès lors, peu de motivation chez l'apprenant.

Un troisième objectif du projet MAI consiste à développer une application de référence servant de cahier des charges et de test au produit informatique. C'est pourquoi l'équipe d'informaticiens a décidé de s'adjoindre des chercheurs du Département de Biologie de la Faculté des Sciences afin de se faire épauler lors de la phase d'analyse et de spécification, ainsi que pendant la validation et la vérification de l'outil informatique. Ainsi, les nouveaux membres de l'équipe MAI ont participé à l'étape de définition des besoins et ont créé des prototypes de modules de cours multimédia qui serviront à tester le produit final. Le cours auquel il a été décidé d'appliquer le système auteur est un cours de génie génétique moléculaire, branche de

la biologie. Le choix de ce cours n'est pas arbitraire. En effet, cette matière est actuellement en pleine expansion et fait l'objet de nombreuses controverses : séquençage des génomes, clonage... De plus, le besoin de formation dans ce domaine est réel, tant au niveau du grand public qu'au niveau des entreprises qui organisent très souvent des recyclages pour leurs chercheurs. Le système MAI s'est donc développé selon deux volets : le volet biologie et le volet informatique. Le volet biologie concerne la tâche de transposition du cours de génétique sur Internet, au moyen de textes mais aussi d'illustrations, de simulations, de séquences vidéo, d'animations en couleurs... De l'autre côté, le volet informatique a pour mission la conception d'un outil permettant la création, l'étude et le stockage de modules indépendants (appelés "unités d'apprentissage"), la construction et la consultation de glossaires, l'élaboration et l'utilisation de questionnaires d'évaluation, la réalisation de cartes de navigation illustrant des parcours sémantiques...

Enfin, il nous apparaît utile de préciser que le projet MAI est encore, actuellement, en stade de développement. Certaines fonctionnalités sont terminées, alors que d'autres sont en cours de réalisation. Par ailleurs, l'intégration des volets biologie et informatique n'a pas encore été réalisée. Par conséquent, tous les contenus servant à tester l'outil informatique, tant au niveau des cours que de la base de données, sont composés, pour l'instant, de données factices. Il en va de même pour le glossaire.

3.2. Le volet biologie

3.2.1. Structure d'un cours interactif

Un cours interactif est divisé en une série de problématiques ou "objectifs". L'apprentissage d'un cours se fait par l'étude des différents objectifs inhérents à ce cours. Chaque objectif regroupe un ensemble de concepts appelés "unités d'apprentissage". Ces unités d'apprentissage, que nous noterons UA dans la suite de cet exposé, sont composées d'un certain nombre de pages et sont organisées dans un réseau sémantique, illustré par une carte de navigation.

3.2.1.1. Le réseau sémantique

L'auteur d'un cours élabore un réseau sémantique organisant les UA les unes par rapport aux autres. Ce réseau permet à l'élève de progresser d'une UA à une autre, selon un parcours suggéré par le professeur. Le réseau sémantique est mis à la disposition de l'élève pendant son apprentissage, sous la forme d'une carte de navigation[12], afin de lui proposer un aperçu de son parcours. Dans cette carte de navigation, les UA déjà apprises sont différenciées de celles à apprendre. Cette carte permet à l'élève de prendre connaissance des UA prérequises à une UA donnée et de choisir celle qu'il veut étudier.

Concrètement, chaque nœud de la carte de navigation représente une unité d'apprentissage. Le nœud situé en bas de la carte de navigation est appelé "nœud objectif" et symbolise l'objectif final du cours. Pour l'atteindre, il faut passer par différents nœuds intermédiaires appelés "nœuds critiques". Ainsi, tout chemin passant par certains nœuds critiques et reliant deux nœuds - dont le deuxième peut être le nœud objectif - illustre un parcours sémantique d'apprentissage.

Chaque nœud possède un questionnaire d'évaluation. Dès lors, les différentes évaluations associées aux nœuds critiques conduisent à l'évaluation finale contenue dans le nœud objectif. L'évaluation globale du cours est l'aboutissement logique de cette série d'évaluations.

[12] Le lecteur trouvera un exemple de cette carte de navigation à la figure 3.1.

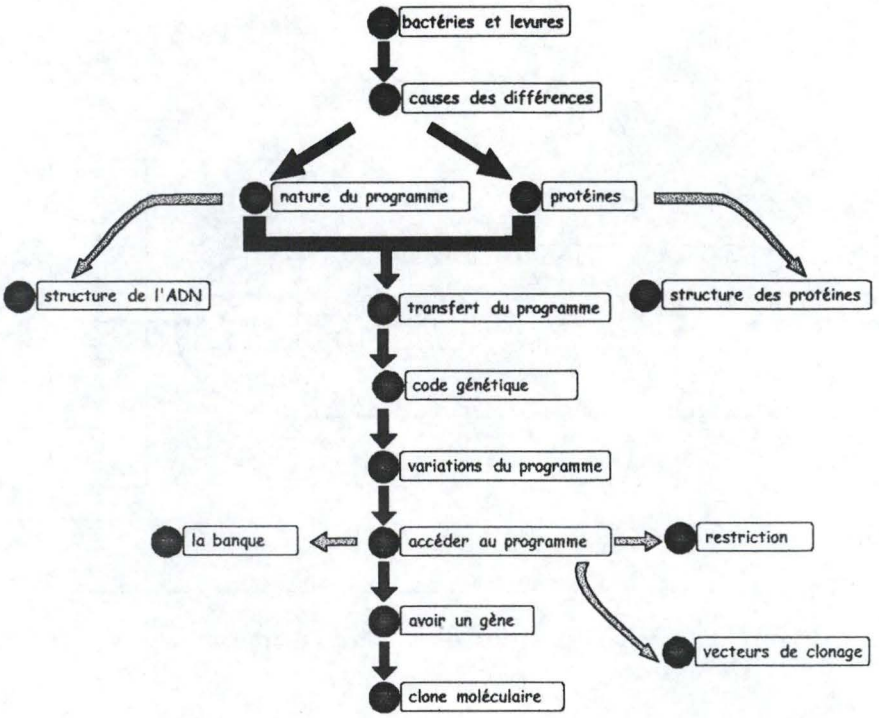


Figure 3.1 : Exemple de carte de navigation

3.2.1.2. L'unité d'apprentissage

Le système auteur MAI se base sur la notion de "concept". En effet, chaque UA matérialise un concept qu'un professeur désire inculquer à ses élèves. Une UA[13] est décrite dans une suite de pages HTML qui rassemblent tous les textes, animations, simulations... ayant trait à un même concept théorique.

Une UA se compose :

- d'un **noyau** constitué d'une suite linéaire de pages qui contiennent les textes présentant le concept concerné et,
- de **ressources périphériques**, autour de ce noyau, qui sont les informations multimédias (schémas, illustrations, animations, simulations, exercices...). Tous ces éléments sont schématisés dans la figure 3.2.

[13] Voir annexe D : Exemple d'une UA réalisée par l'équipe de biologistes.

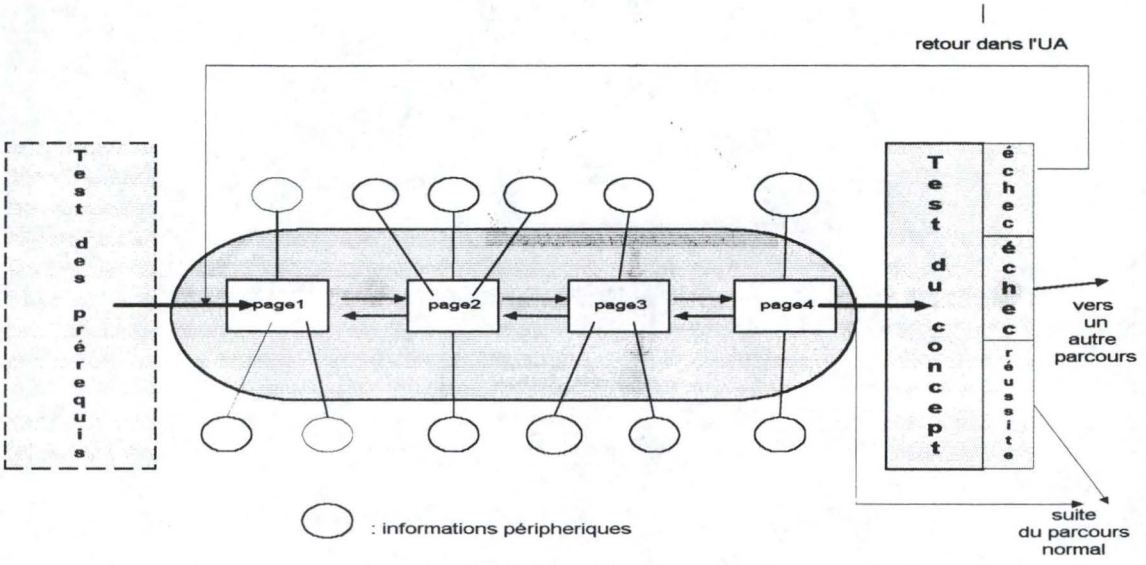


Figure 3.2 : Schématisation d'une unité d'apprentissage

Une UA présente trois caractéristiques importantes :

- Elle ne se rapporte qu'à un et un seul concept ;
- Elle a été conçue par un seul auteur ;
- Un numéro de version peut lui être attribué, ce qui permet de conserver un certain historique de la création de cette UA.

Une UA est une ressource partagée avec toute la communauté des professeurs bien qu'elle reste propriété de son auteur qui peut la faire évoluer comme bon lui semble. Néanmoins, elle peut également être modifiée par un professeur qui n'en est pas l'auteur. Dans ce cas, elle devient une nouvelle UA.

L'apprentissage d'une UA aboutit à un questionnaire d'évaluation que doit remplir l'élève. Cette évaluation vise uniquement à indiquer à l'apprenant son niveau de compréhension de l'UA étudiée et ne cherche nullement à lui interdire l'accès à l'UA suivante.

Le scénario d'apprentissage d'un cours se déroule, en réalité, selon quatre niveaux. L'apprenant choisit tout d'abord un cours. Il se trouve alors au niveau 1. Ensuite, s'il décide d'accéder aux différentes UA du cours choisi, il se retrouve au niveau 2. Dès qu'il choisit de consulter une UA appartenant à ce cours, il atteint le niveau 3. Enfin, il peut apprendre les pages de cette UA qui constituent le niveau 4.

La figure 3.3 illustre les fonctionnalités proposées à chaque niveau de l'apprentissage (sous la forme d'onglets sur l'interface). Cependant, des scénarios plus détaillés sont présentés dans le point suivant.

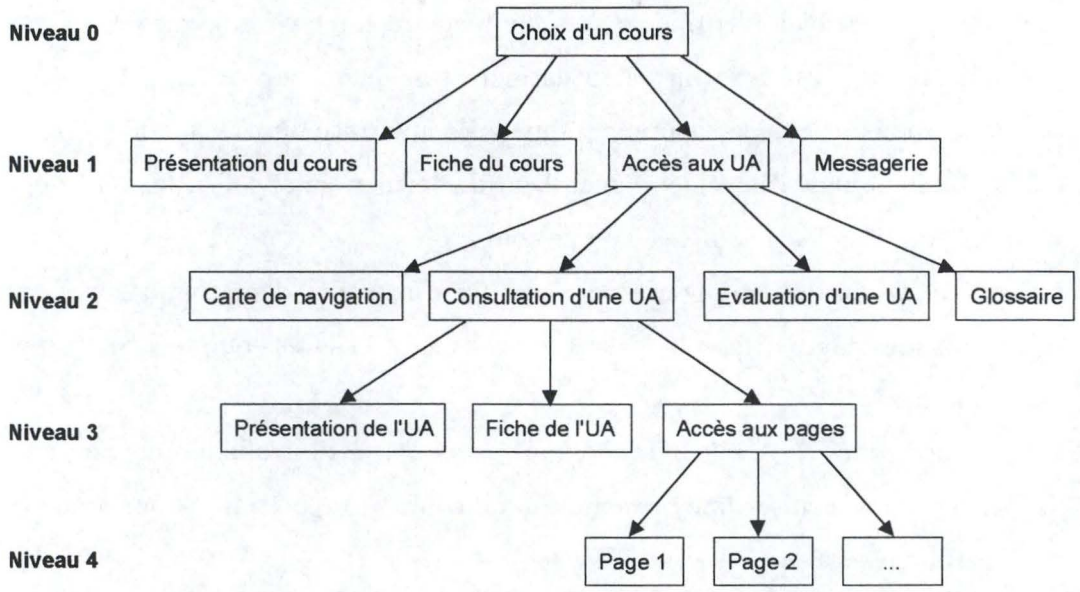


Figure 3.3 : Les quatre niveaux du scénario de l'apprentissage d'un cours

3.3. Le volet informatique

3.3.1. Le scénario de l'apprenant

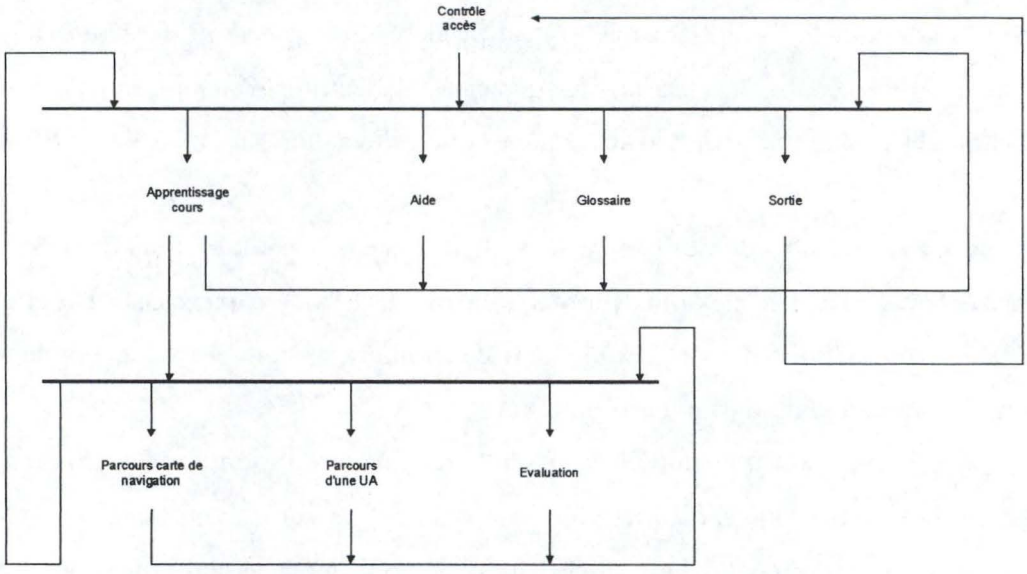


Figure 3.4 : Scénario de l'apprenant

1) Si l'étudiant accédant au système auteur MAI y est déjà inscrit, il doit s'identifier en écrivant dans les champs appropriés son nom, son mot de passe, ainsi que sa catégorie ("Apprenant").
2) S'il n'est pas encore inscrit, il est invité à le faire.
3) Si l'apprenant est bien identifié, le système restaure son environnement de travail tel qu'il l'avait quitté à la fin de sa dernière session. Pour ce faire, le système conserve un historique de l'apprenant qui garde une trace du travail effectué et qui comprend le nom de l'élève, le nom et l'état du dernier cours consulté, ainsi que le nom et l'état des UA consultées dans ce cours.
4) La session de travail commence par le choix d'un cours dans une arborescence listant tous les cours disponibles dans le système MAI. Ces cours sont regroupés par domaines.
5) Dès qu'un cours est sélectionné, une page d'accueil s'affiche et permet à l'apprenant de visualiser une présentation du cours et de prendre connaissance de son intitulé général.
6) Dès ce moment, toutes les fonctions destinées à l'étude d'un cours sont activées dans les onglets de l'interface.
7) Dans l'onglet *Carte*, l'élève a accès à une carte de navigation présentant l'organisation des différentes UA nécessaires à l'apprentissage de l'objectif retenu. Bien sûr, il n'y a accès qu'en mode consultation. Cette carte est la matérialisation du réseau sémantique déterminé par le professeur.
8) Après que l'apprenant a sélectionné une UA dans la carte de navigation, un écran de travail, dont le format ("canevas") est prédéterminé par le professeur, s'affiche dans l'onglet *Unité d'Apprentissage* et lui permet de débuter son apprentissage.
9) Chaque UA se compose de plusieurs pages que l'élève doit parcourir d'une manière linéaire.
10) L'apprentissage est interactif en ce sens que l'apprenant peut, en plus de parcourir les textes et visualiser les illustrations, s'entraîner à l'aide d'exercices. En effet, il a le pouvoir de contrôler le déroulement de simulations comme, par exemple, une simulation de manipulation dans un laboratoire.
11) Dès que l'apprenant a terminé l'étude d'une UA, un questionnaire d'évaluation lui est proposé dans l'onglet *Evaluation*. Le résultat de cette évaluation est stocké dans la base de données. Il est alors disponible pour le professeur concerné.

12) Dans l'onglet *Glossaire,* l'élève a la possibilité d'obtenir des renseignements complémentaires au sujet d'un terme précis. Il a été convenu que le glossaire proposé est limité aux termes appartenant au cours consulté.

13) L'onglet *Messagerie* offre à l'apprenant la possibilité de réaliser des échanges avec d'autres utilisateurs du système (professeurs ou apprenants).

3.3.2. Le scénario du professeur

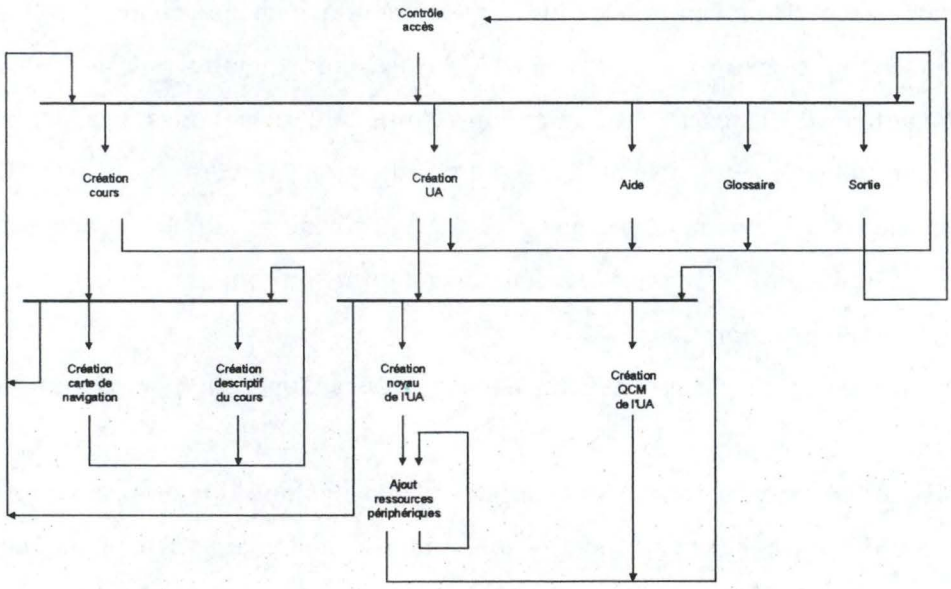


Figure 3.5 : Scénario du professeur

1) Tout comme l'apprenant, le professeur doit s'identifier dès son arrivée dans le système MAI. Il doit compléter la fiche d'identification par ses nom, mot de passe et catégorie ("Professeur").

2) Si c'est sa première entrée dans le système, il est invité à s'inscrire sur-le-champ.

3) Si le professeur passe avec succès l'étape d'identification, le système restaure son environnement de travail tel qu'il l'avait quitté à la fin de sa précédente connexion. Pour ce faire, le système conserve un historique du professeur qui garde une trace du travail effectué et qui comprend le nom du professeur, le nom et l'état (fini ou pas fini) du cours sur lequel il a travaillé la dernière fois et le nom et l'état (ouverte ou fermée) des UA associées à ce cours.

4) Contrairement à l'apprenant, le professeur a le choix, à ce stade, entre :

- sélectionner un cours existant dans l'arborescence de présentation des cours, ce qui entraînera l'affichage d'une présentation du cours choisi, suivie de son intitulé général ;
- créer un nouveau cours.

5) S'il décide de consulter un cours, toutes les fonctions dont il a besoin pour travailler sur ce cours sont instantanément activées dans les onglets. Ces fonctions sont plus nombreuses que dans le cas de l'apprenant car elles comprennent les procédures de création et de modification d'un cours, d'une carte de navigation, d'une UA... De plus, il a été établi que chaque élément créé par un professeur et stocké dans le système MAI, est "propriétaire". C'est–à-dire que seul l'auteur d'une carte de navigation, d'une UA, d'un cours... a le droit de modifier cet élément. Néanmoins, la réutilisation de ces objets par d'autres intervenants est vivement encouragée. La réutilisation, si elle est accompagnée d'une modification, entraîne la naissance d'un tout nouvel objet ayant les caractéristiques du premier.

6) Si le professeur décide de créer un nouveau cours, l'opération se déroule comme suit :
   - Il doit commencer par créer une fiche d'identification du cours qui contient le titre du cours, l'identification de son auteur et le domaine dans lequel ce cours est créé ;
   - Il s'agit ensuite de réaliser un descriptif de ce cours qui comprend une présentation du cours, une indication de la charge horaire... ;
   - Le professeur crée la carte de navigation et y associe les UA qu'il veut intégrer à son cours. Il hiérarchise ces UA entre elles selon les objectifs du cours. Cette carte de navigation comporte des informations sur les nœuds critiques de la carte, la liste des UA qu'il faut parcourir pour arriver à ces nœuds critiques (appelées "UA critiques") et le nom de l'UA objectif de ce cours ;
   - Il conçoit finalement le questionnaire d'évaluation du cours qui permettra à chaque étudiant d'évaluer son niveau d'apprentissage de tous les concepts associés à ce cours ;

7) Dans l'onglet *Carte*, le professeur a accès en construction, en modification ou en consultation (selon que cette carte lui appartient ou non) à la carte de navigation du cours qu'il a choisi. Dans les deux premiers cas, le professeur peut relier à la carte de navigation des UA existantes (lui appartenant ou pas) pour former le réseau sémantique désiré.

8) L'onglet *Unité d'apprentissage* permet au professeur de consulter les UA appartenant au cours qu'il a choisi ou de les modifier s'il en est l'auteur. Il peut aussi en créer de nouvelles. Les étapes de la création d'une UA sont les suivantes :
   - Il crée la fiche d'identification de l'UA. Celle-ci doit contenir le titre de l'UA, l'identification de son auteur et le domaine dans lequel elle est créée ;
   - Il rédige ensuite un descriptif de cette UA qui comporte une présentation de l'UA, l'indication de la charge horaire au niveau de l'apprentissage... ;
   - Il choisit alors un canevas. Un canevas est une page à cadres où chaque cadre peut avoir une taille et un contenu différents. L'organisation des cadres sur la page représente le canevas. La figure 3.6 illustre les différents canevas que propose le système MAI. Le nombre de zones est limité à 3 pour des raisons d'ergonomie de l'interface. Le professeur peut créer un nouveau canevas si aucun de ceux qui lui sont proposés ne lui convient ;

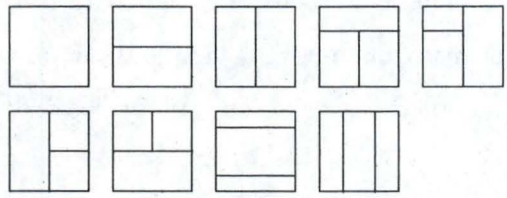


Figure 3.6 : Canevas proposés par le système MAI

   - Il paramètre le canevas désiré en fixant la taille et la destination des différentes zones, la visibilité ou non des bordures entre les zones, ainsi que leur couleur, le redimensionnement ou non des zones et la possibilité de faire défiler la page ;
   - Le professeur construit le noyau de l'UA qui se compose de pages de base et de leur ordonnancement. Une barre de navigation permettra à l'apprenant de parcourir les pages de l'UA ;
   - Il crée enfin le questionnaire d'évaluation de l'UA qui lui permettra d'évaluer le niveau d'apprentissage de l'étudiant.

9) Dans l'onglet *Evaluation*, le professeur peut consulter le questionnaire d'évaluation proposé pour chaque UA ou bien en créer un nouveau. Il en va de même pour le questionnaire d'évaluation du cours.

10) Il peut, à tout moment, dans l'onglet *Glossaire*, consulter le glossaire associé au cours sur lequel il travaille ou en créer un nouveau si ce glossaire est vide. Il peut

également importer un glossaire existant provenant d'un autre cours et l'adapter par la suite aux caractéristiques du cours effectif.

11) Enfin, grâce aux fonctionnalités disponibles dans l'onglet *Messagerie*, le professeur a la possibilité d'envoyer à ses élèves des réactions, des explications ou des conseils par courrier électronique au vu des performances de ces derniers.

3.3.3. Les technologies employées

Le système auteur du projet MAI fonctionne dans un environnement distribué. Partant, il est implémenté entièrement en Java selon une architecture client – serveur. Comme il se doit, il a été conçu de manière à alléger au maximum le travail de la station cliente en lui assignant les aspects de présentation et le traitement de petites applications. Le serveur, de son côté, traite les applications plus complexes et joue le rôle de dispositif de stockage de données. Les applications distribuées de l'architecture client – serveur sont mises en œuvre grâce à RMI qui permet à des objets Java d'appeler des méthodes d'objets Java s'exécutant sur des ordinateurs distants.

Pour le cours de génie génétique auquel s'applique le système auteur de MAI, de nombreuses simulations d'expériences ont été élaborées au moyen du logiciel d'animation Flash 3 de Macromedia et d'un outil de dessin appelé Freehand 8. Pour communiquer avec la base de données, on utilise la technologie JDBC-ODBC offerte par Java. Cette technologie a fait l'objet d'un long développement dans le premier chapitre de ce travail.

3.3.3.1. L'architecture client – serveur

Comme son nom l'indique, le modèle client – serveur se compose d'un client et d'un serveur. Ces deux entités se situent dans des espaces d'adressage différents. Le client est, selon [VOSS 98], "un ordinateur ou un programme qui, à l'intérieur d'un réseau, demande des services à un serveur". D'après ce même auteur, le serveur est "un ordinateur dédié à l'administration d'un réseau informatique. Il gère l'accès aux ressources et aux périphériques et les connexions des différents utilisateurs. Il est équipé d'un logiciel de gestion du réseau : un serveur de fichiers prépare la place mémoire pour des fichiers, un serveur d'impression gère et exécute les sorties sur les

imprimantes du réseau, enfin un serveur d'applications rend disponibles sur son disque dur les programmes pouvant être appelés à travers le réseau". Une telle architecture permet de répartir des objets sur un réseau et d'activer leurs méthodes à distance. En résumé, l'application cliente envoie des requêtes sur des objets se trouvant sur le serveur distant et l'application serveur fait fonction de structure d'accueil des objets et d'exécution des opérations. Pour pouvoir accéder aux objets distants, la station cliente ne doit pas nécessairement connaître leur localisation. Seule l'interface de l'objet distant est requise du côté client. Elle renferme le nom, les opérations et les attributs associés à cet objet. Dans le langage de programmation Java, une interface est définie, selon [LEMAY-CADENHEAD 00], comme un modèle de comportement censé être implémenté par d'autres classes. Concrètement, une interface ne fournit que des définitions de méthode abstraites qui doivent être implémentées dans une classe, en utilisant les mêmes signatures de méthode que celles employées dans l'interface. Malgré leurs définitions divergentes, une interface et une classe présentent un grand nombre de similitudes. En effet, elles doivent toutes deux être déclarées dans des fichiers source `.java` et elles sont compilées de la même manière en fichiers `.class`. L'une des seules différences entre une classe et une interface repose sur le fait que cette dernière ne peut être instanciée : l'opérateur `new` ne peut créer qu'une instance de classe.

3.3.3.2. La technologie RMI

Dans un système distribué, des applications fonctionnant dans des espaces d'adressage différents doivent parvenir à communiquer. Pour atteindre cet objectif, de nombreux développeurs utilisent les "sockets". Les sockets sont les points de connexion situés à chaque extrémité d'une connexion (TCP, UDP...). Il existe un socket du côté émetteur et un autre du côté récepteur. Un socket TCP est composé de l'adresse IP de l'ordinateur hôte et du port TCP auquel ce dernier écoute. La solution qui consiste à utiliser des sockets pour ouvrir, maintenir et fermer une connexion entre deux hôtes distants nécessite le partage entre l'hôte émetteur et l'hôte récepteur de protocoles d'encodage et de décodage des messages échangés. C'est une solution flexible et suffisante pour des communications simples et occasionnelles. Cependant, elle est assez rudimentaire, lourde à déployer et propice aux erreurs. C'est pourquoi la technologie RMI est de plus en plus souvent utilisée dans des environnements d'applications exclusivement Java. RMI signifie "Remote Method Invocation". C'est

un ensemble de protocoles qui permettent aux objets Java de communiquer à distance avec d'autres objets Java. RMI est un protocole relativement simple qui ne travaille qu'avec des objets Java, contrairement à d'autres protocoles plus complexes comme CORBA. Cette technique fournit une interface au travers de laquelle le serveur et un client peuvent dialoguer. Cette interface définit l'ensemble des méthodes supportées par le serveur et constitue la base d'appel pour les clients. Elle est implémentée dans un objet se trouvant uniquement sur le serveur. Ainsi, un client ne pourra faire appel qu'à des méthodes définies dans l'interface, même si le serveur définit d'autres méthodes (non destinées aux clients). Dans les systèmes distribués utilisant RMI, il existe deux types d'objets particuliers et créés automatiquement par le compilateur *rmic* : l'un situé sur le client, et l'autre, sur le serveur. L'objet local est appelé "stub" et est destiné à traiter l'invocation d'une méthode d'un objet distant. Le client, qui héberge ce stub, connaît l'ensemble des méthodes de l'objet distant qu'il veut appeler, grâce à l'interface décrite dans le point précédent. Concrètement, le stub délègue tous les appels qu'il reçoit à l'objet distant correspondant. L'objet situé sur le serveur s'appelle, quant à lui, "skeleton" et a pour objectif de recevoir les appels envoyés par le stub. Pour chaque appel reçu, il invoque la méthode effective sur l'objet local au serveur et envoie le résultat de cette méthode au stub situé sur l'ordinateur client. Cette alternative aux sockets procure au développeur l'illusion d'appeler une procédure locale, alors qu'en réalité, son appel est envoyé à la méthode cible distante.

3.3.4. Les langages de programmation utilisés

Le système MAI est conçu au moyen des langages Java et HTML. Dans les points suivants, nous allons aborder chacun de ces langages et décrire leur utilité dans le projet MAI.

3.3.4.1. Le langage HTML

Le langage HTML ("HyperText Markup Language") est utilisé pour la création des cours interactifs et la présentation de leurs simulations et illustrations. Bien entendu, le professeur auteur ne doit pas nécessairement connaître ce langage pour créer ses UA. Un éditeur HTML développé en Java lui permettra de composer ses textes et de disposer ses illustrations, simulations d'expériences, sons... de la même manière qu'un traitement de texte classique.

3.3.4.2. Le langage Java

Le système MAI s'inscrivant dans le domaine du World Wide Web, Java apparaît comme la technologie multimédia par excellence. En effet, Java permet de créer des applets, qui sont, comme nous l'avons dit précédemment, des programmes pouvant être lus par tous les navigateurs de réseau ("browsers"). Par ailleurs, Java est un langage "orienté objet" qui permet aux programmeurs de définir des objets au moyen de leurs données et de leurs opérations (services qu'on peut leur demander). Cette technique s'appelle l'encapsulation. D'autres techniques comme l'héritage et le polymorphisme autorisent les programmeurs à organiser leurs objets dans des hiérarchies, ce qui permet à certains objets d'hériter des données et opérations d'autres objets. Cette structure en objets offre la possibilité aux programmeurs de ré-utiliser des objets créés par des collègues dans un souci d'économie. De fait, des problèmes traités dans des contextes différents peuvent très bien posséder, dans une large mesure, des caractéristiques similaires. Certains d'entre eux peuvent, dès lors, être résolus par les opérations d'un même objet, grâce à la technique de l'héritage, ou par des opérations dérivées de celles-là, par la technique du polymorphisme. De plus, la technologie RMI que propose Java facilite la mise en place et l'utilisation de l'architecture client – serveur.

3.3.5. Le modèle client – serveur du système MAI

L'ordinateur client de l'architecture client – serveur déployée est appelé "front-end". L'ordinateur serveur est appelé, quant à lui, "back-end". Une gestion des cartes de navigation, des questionnaires d'évaluation, des glossaires, et des UA est offerte par des programmes au niveau serveur (back-end), de même qu'au niveau client (front-end). Dès qu'un objet (UA, glossaire, carte de navigation ou évaluation) est créé par un professeur, le client (front-end) transfère les informations le concernant et l'ensemble des fichiers reliés à cet objet vers le serveur (back-end) qui va les stocker dans une base de données gérée par un SGBD de type SQL Server. Les UA, questionnaires, cartes de navigation et glossaires à peine créés sont alors mis à disposition des autres professeurs dans un but de partage et de réutilisation.

3.3.5.1. Le niveau front-end

Une procédure de contrôle d'accès limite l'entrée dans le système aux seules personnes autorisées, grâce à une procédure d'identification (nom d'utilisateur et mot de passe). Cette procédure permet, en outre, de déterminer le type d'interface à présenter et les actions autorisées en fonction du profil de l'utilisateur identifié. En effet, il existe deux profils distincts d'utilisateurs qui ont des privilèges différents :

- les **professeurs** se voient attribuer une interface permettant un accès en lecture aux cours dont ils ne sont pas les auteurs et un contrôle total sur leurs propres cours. Comme nous l'avons dit plus haut, un professeur peut modifier un objet ne lui appartenant pas. Il est autorisé à le faire car l'objet sur lequel il travaille est une copie de l'objet initial qui deviendra un nouvel objet à part entière lors de l'enregistrement des modifications.

- les **apprenants** ne disposent que d'un accès en lecture aux cours disponibles.

A ce même niveau, il est nécessaire de disposer d'interfaces adéquates permettant aux utilisateurs d'accéder au système MAI et d'accomplir leur travail de manière optimale et ergonomique. En fait, six dossiers sont disponibles à tout moment dans l'interface générale, grâce à des onglets et à une arborescence de présentation des cours. Un simple clic sur un onglet ou dans l'arborescence permet à l'utilisateur de voyager d'une fonction à l'autre.

Ces six fonctions sont :

- Travailler sur un cours (Arborescence de présentation des cours) ;
- Naviguer dans la carte de navigation (Onglet *Carte*) ;
- Travailler sur une UA (Onglet *Unité d'Apprentissage*) ;
- Accéder au module d'évaluation (Onglet *Evaluation*) ;
- Consulter le glossaire (Onglet *Glossaire*) ;
- Envoyer un message électronique à un autre utilisateur du système (Onglet *Messagerie*).

Le niveau front-end dispose de nombreuses procédures qui ont pour objectif de réaliser tous les désirs exprimés directement ou indirectement par l'utilisateur. Des procédures correspondent aux six fonctionnalités ci-dessus mais il en existe beaucoup d'autres. Les procédures les plus importantes sont le traitement d'un cours et le traitement d'une UA. Nous en développons quelques-unes dans les points suivants.

a) Procédure de traitement d'un cours

Du point de vue du professeur, travailler sur un cours consiste à créer ou modifier un cours. Les fonctionnalités de la création d'un cours sont les suivantes :

- créer la fiche d'identification du cours ;
- créer un descriptif du cours ;
- créer la carte sémantique de navigation ;
- créer le questionnaire d'évaluation du cours ;
- sauvegarder le cours créé.

Du point de vue de l'apprenant, travailler sur un cours correspond à étudier son contenu et à s'auto-évaluer dès la fin de cet apprentissage.

b) Procédure de traitement d'une unité d'apprentissage

Travailler sur une UA consiste, pour le professeur, à créer ou modifier une UA, concevoir les exercices à insérer dans cette UA, lui ajouter des objets multimédias et réaliser le questionnaire d'évaluation lié à cette UA. Il existe également des procédures qui transfèrent une UA du client vers le serveur et qui récupèrent une UA du serveur vers le client. Les fonctionnalités de la création d'une UA sont les suivantes :

- créer la fiche d'identification de l'UA ;
- créer le descriptif de l'UA ;
- choisir un canevas ;
- paramétrer le canevas désiré ;
- modifier le canevas d'une UA ;
- créer le noyau de l'UA ;

- créer le questionnaire d'évaluation de l'UA ;
- sauvegarder l'UA créée.

Pour l'apprenant, travailler sur une UA consiste à consulter ses différentes pages de manière linéaire et à s'auto-évaluer dès la fin de l'apprentissage de l'UA.

c) Procédure d'aide

Une autre procédure front-end fournit une option d'aide à l'utilisateur sous la forme d'un guide explicatif du fonctionnement de l'outil. Elle contient également un répertoire des différentes questions déjà posées et des solutions qui y ont été apportées. L'aide est composée de fichiers HTML possédant des liens hypertextes se référençant les uns les autres.

d) Procédure du glossaire

Un glossaire est mis à la disposition de l'utilisateur. Ce glossaire donne rapidement des informations complémentaires au sujet d'un mot choisi parmi quelques deux cents termes de biologie moléculaire. Il est entendu que, malgré son accessibilité permanente, le glossaire est restreint aux termes appartenant au cours dans lequel l'utilisateur travaille. Cette procédure sera développée plus largement dans le chapitre suivant.

e) Procédure de contrôle des sorties

Une autre procédure a pour rôle la gestion de la sortie d'utilisateur du système, en veillant à enregistrer le travail qui a été réalisé. Grâce à la sauvegarde de son historique, l'utilisateur, qu'il soit apprenant ou professeur, verra son environnement de travail restauré lors de sa prochaine connexion.

3.3.5.2. Le niveau back-end

Au niveau du serveur, se trouvent la base de données ainsi que de nombreux programmes qui jouent un rôle important lors des transactions client – serveur : expédier au client le nécessaire pour la réalisation de son travail, que ce soit un outil de

création de cours et d'UA pour le professeur ou une aide dans l'apprentissage d'un cours pour l'élève. Toutes les communications entre le serveur et le client sont assurées par la technologie RMI.

La base de données est une base de données relationnelle, c'est-à-dire qu'elle se compose de tables reliées les unes aux autres au moyen de contraintes référentielles. Tous les accès à la base de données sont effectués au niveau back-end pour des raisons de sécurité. En effet, dans une architecture client - serveur, il est interdit à une station cliente d'accéder à des ressources se trouvant dans un autre espace d'adressage. Pour ce faire, elle doit impérativement faire appel à une méthode se trouvant sur le serveur et qui exécutera sa requête.

Plusieurs procédures effectuent des opérations sur les données des UA, telles que la lecture d'informations d'une UA particulière, la copie du contenu d'une UA, le transfert des informations de la base de données vers la station client et vice versa...

3.3.6. Le stockage des données : Dictionnaire des données

Le serveur possède un dispositif de stockage et de gestion des données robuste et persistant. Il s'agit, comme nous l'avons déjà énoncé ci-dessus, d'une base de données relationnelle gérée par un SGBD de type SQL Server et qui se trouve sur le serveur. Chaque nouvel objet créé par un professeur, l'historique de chaque apprenant, de même que les résultats qu'il a obtenus aux divers tests d'évaluation sont stockés dans la base de données. Les principaux types d'entité de cette base de données sont décrits ci-dessous. Le schéma conceptuel de la base de données du système MAI, ainsi que le schéma logique implémentant ce schéma conceptuel peuvent être consultés aux annexes E et F.

a) Apprenant

L'apprenant se définit comme l'acteur du système qui utilise celui-ci pour apprendre un objectif précis en consultant une série d'UA. Ces UA ont été écrites par

un professeur dans le cadre d'un cours et interviennent dans un réseau sémantique. Un Historique_Apprenant[14] lui est associé.

Chaque **apprenant** possède les attributs suivants :
(Les attributs soulignés forment l'identifiant primaire.)

Nom
Prénom
Adresse
Pays
Faculté
Université[15]

b) Professeur

Il s'agit de l'acteur qui alimente le système. Il l'utilise pour créer ou modifier des cours et des UA. Un cours doit atteindre un objectif précis. Pour atteindre cet objectif, un professeur relie entre elles un certain nombre d'UA sous la forme d'un réseau sémantique. Ces UA interviennent dans l'apprentissage d'un cours par un apprenant. Un Historique_Professeur[16] lui est associé.

Chaque **professeur** possède les attributs suivants :
(Les attributs soulignés forment l'identifiant primaire.)

Nom
Prénom
Faculté
Université[17]

c) Cours

Un cours est construit ou modifié par un professeur. Un cours est composé d'UA autonomes que le professeur peut relier les unes aux autres dans un réseau sémantique. Un questionnaire d'évaluation lui est également associé. Il est consulté par un apprenant lors de son apprentissage. Il comprend une série de nœuds intermédiaires ("nœuds critiques") et vise l'atteinte d'un objectif précis ("nœud objectif").

---

[14] Cet objet est expliqué ci-dessous.
[15] L'université correspond au domaine auquel l'apprenant appartient.
[16] L'objet Historique_Professeur est expliqué ci-dessous.
[17] L'université correspond au domaine auquel le professeur appartient.

Chaque **cours** possède les attributs suivants :
(Les attributs soulignés forment l'identifiant primaire.)

Id-Faculté[18]
Titre_Cours
Nom_Auteur
Date[19]
Université

D'autres attributs forment une fiche pédagogique contenant des informations sur le cours et une fiche de présentation du cours.

d) Nœud

Un nœud est une partie du réseau sémantique qui hiérarchise les UA dans la poursuite d'un objectif. Chaque nœud référence une UA. Dans une même carte sémantique, il existe des nœuds critiques qui représentent des objectifs intermédiaires et un seul nœud objectif qui correspond à l'objectif général à atteindre. A chaque nœud est associé un questionnaire d'évaluation.

Chaque **nœud** possède les attributs suivants :
(Les attributs soulignés forment l'identifiant primaire.)

Titre_UA
Version_UA[20]
Id_Faculte
Sorte

Chaque **nœud objectif** possède les attributs suivants :
(Les attributs soulignés forment l'identifiant primaire.)

Nom_Objectif
Nom_Cours

e) Unité d'apprentissage

Une UA est l'ensemble des données relatives à un concept qu'un professeur stocke dans le système. Elle est créée dans le cadre d'un cours et concourt à la réalisation d'un objectif précis. Elle peut être insérée dans un réseau sémantique d'apprentissage. Chaque UA comprend une information de base constituée d'une succession de pages de

[18] Id_Faculté est l'identifiant du cours suivant la convention adoptée aux FUNDP.
[19] Date contient la date de création du cours.

base, un certain nombre de ressources périphériques ayant pour objectif de faciliter l'assimilation de l'information de base, un questionnaire d'auto-évaluation et des objets multimédias.

Chaque **unité d'apprentissage** possède les attributs suivants :
(Les attributs soulignés forment l'identifiant primaire.)

Titre
Version
Présentation
DateCréation
Statut[21]

f) Page

La page est le composant minimal de l'information de base d'une UA. Des périphériques peuvent lui être associés.

Chaque **page** possède les attributs suivants :
(Les attributs soulignés forment l'identifiant primaire.)

TitreUA
VersionUA
N°Page
TitrePage
Contenu

g) Périphérique

La ressource périphérique est le composant minimal de l'information périphérique d'une UA. Il sert à faciliter l'assimilation de l'information de base. Chaque périphérique a un type (image, son, vidéo, tableau...). En réalité, un objet périphérique (inséré dans une page d'une UA) ne possède qu'une référence vers une ressource contenue dans la table des ressources. De cette manière, les ressources ne sont jamais dupliquées dans la BD.

---

[20] Un nœud étant la représentation dans la carte sémantique d'une UA, il est normal qu'il possède le même identifiant.
[21] Le statut d'une UA représente son état de finition.

Chaque **périphérique** possède les attributs suivants :
(Les attributs soulignés forment l'identifiant primaire.)

Titre
N°Page
TitreUA
VersionUA
Contenu
Endroit[22]

h) Question

Toutes les questions qu'un professeur associe à un concept en vue d'interroger l'apprenant et d'évaluer son niveau d'apprentissage forment le questionnaire d'auto-évaluation. Chaque question se compose du corps de la question et de la réponse correcte.

Chaque **question** possède les attributs suivants :
(Les attributs soulignés forment l'identifiant primaire.)

Id
TitreUA
VersionUA
Enoncé

i) Historique_Apprenant

L'historique de l'apprenant permet à l'administrateur du système MAI de recréer le dernier environnement de travail de l'apprenant. Cette fonction est particulièrement utile lorsque l'apprenant désire étaler son apprentissage sur une longue période. Chaque historique stocke également des informations sur l'identification de l'apprenant, le dernier cours sur lequel cet apprenant a travaillé (nom et état – fini ou pas -), la liste des UA de ce cours déjà consultées (noms et états – parcourues ou pas -), les cotes obtenues lors de l'évaluation des cours et UA.

Chaque **Historique_Apprenant** possède l'attribut suivant :

NomHistorique

[22] Endroit signifie la zone du canevas où ce périphérique doit s'afficher.

j) Historique_Professeur

L'historique du professeur contient les informations sur le dernier environnement de travail du professeur de manière à pouvoir le restaurer lors de la prochaine connexion. Cet historique contient le nom et l'état (fini ou pas) du cours que le professeur était en train de créer lorsqu'il a mis fin à sa dernière session de travail. Il comprend également une liste de tous les noms et états (création finie ou pas) des UA associées à ce cours.

Chaque **Historique_Professeur** possède l'attribut suivant :

NomHistorique

3.4. Le glossaire imaginé par l'équipe MAI

En 1999, certains membres de l'équipe MAI ont mené une recherche sur Internet dans le but de trouver et comparer des glossaires et lexiques relatifs à la biologie moléculaire. A partir de la vingtaine de sites intéressants qu'ils ont consultés, il ont recensé deux cents termes courants utilisés en biologie moléculaire. Ils les ont classés selon l'ordre alphabétique et puis les ont numérotés de 001 à 200. Ces deux cents termes ont ensuite été catalogués dans un fichier File Maker Pro qui comporte deux cents fiches, chacune se rapportant à un terme.

Chaque terme est défini par :

- son numéro ;
- son terme en français ;
- sa traduction en anglais ;
- son (ou ses) abréviation(s) éventuelle(s) ;
- son statut ;
- ses synonymes, antonymes et dérivés ;
- sa définition.

Bien entendu, la structure choisie il y a deux ans pour réaliser ce catalogue File Maker Pro ne correspond plus vraiment, ni à la réalité, ni aux nouvelles attentes. Effectivement, une décision fut prise de stocker les termes du glossaire non plus dans un fichier mais dans la base de données relationnelle existante qui contient déjà toutes

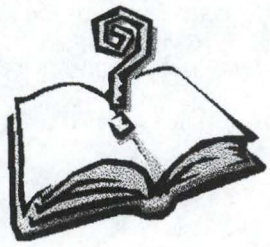
les informations à propos des autres entités du système MAI. Dès lors, les caractéristiques ci-dessus ont été passées en revue lors de réunions. Certaines ont été conservées alors que d'autres ont été supprimées ou restreintes. Ces résolutions ont été prises suite à une confrontation de ces attributs avec les idées dégagées lors des recherches effectuées durant le stage au Devon County Council d'Exeter.

Au cours de ces réunions, il a également été décidé que le glossaire présenterait quatre fonctionnalités :

1) Accès à la définition d'un terme à partir d'une UA, au moyen d'un "ToolTip" ;
2) Accès au glossaire et à tous les attributs d'un terme, à partir du texte d'une UA, au moyen d'un lien hypertexte associé à ce terme ;
3) Accès à la définition et autres attributs d'un terme à partir du glossaire ;
4) Création, chargement ou modification d'un glossaire ;

Les fonctionnalités 1) et 2) n'ont pas été inclues dans ce travail car elles nécessitent, d'une part, la maîtrise du traitement et de l'affichage des UA. D'autre part, l'état d'avancement actuel du développement de cette partie du projet MAI ne permet pas encore d'y intégrer les deux premières fonctionnalités. Néanmoins, des méthodes implantées sur le serveur seront mises à la disposition de l'équipe MAI dans le but de faciliter au maximum le déploiement de ces deux procédures. La création du glossaire fera l'objet des fonctionnalités 3) et 4). Elles s'appliquent respectivement aux interfaces apprenant et professeur. Des descriptions beaucoup plus détaillées des étapes de la réalisation du glossaire sont présentées dans le chapitre suivant.

4
Le glossaire MAI



Ce quatrième chapitre est consacré à l'outil développé dans le cadre de ce travail : le glossaire du système MAI. Le lecteur y trouvera une description et des spécifications détaillées. La description du glossaire comprend la démarche d'analyse adoptée, le contexte d'utilisation et les fonctionnalités offertes aux utilisateurs du système MAI. Les spécifications du glossaire sont exposées au moyen des techniques des cas d'utilisation et des diagrammes d'interaction proposés par le langage de modélisation UML. Le chapitre se clôture par une présentation de la partie de la base de données MAI consacrée au glossaire.

4.1. Démarche d'analyse

C'est par des recherches en bibliothèque que nous avons décidé d'entamer l'analyse du logiciel. Il s'agissait de découvrir l'étymologie du terme "glossaire" et d'analyser les définitions proposées par un maximum d'ouvrages. Ces recherches se sont étendues vers des termes synonymes ou apparentés comme, par exemple, glose, dictionnaire, lexique, vocabulaire... Lors de la seconde étape de l'analyse, d'autres investigations ont été effectuées, au moyen d'Internet, dans le but d'examiner le "design" ainsi que les fonctionnalités proposées par une centaine de glossaires "on-line". Les résultats de ces recherches sont présentés dans le chapitre 2.

4.2. Contexte d'utilisation

Il existe deux catégories d'utilisateurs du système MAI : le professeur et l'apprenant. Cette distinction se reflète également dans l'interface et les fonctionnalités du glossaire puisque ce dernier s'adapte à la catégorie de la personne qui l'utilise. Le professeur agit activement sur le glossaire en tant qu'auteur, alors que l'apprenant joue le rôle passif d'examinateur. Ces deux personnages-types évoluent néanmoins dans le même contexte : ils travaillent sur le glossaire d'un cours qui est dispensé dans une faculté universitaire.

4.3. Fonctionnalités du glossaire

Les fonctionnalités du glossaire diffèrent, comme nous l'avons déjà annoncé plus haut, selon la catégorie de l'utilisateur.

4.3.1. Fonctionnalités spécifiques à l'apprenant

Lorsque l'apprenant accède au glossaire, la liste des termes appartenant au glossaire du cours qu'il apprend apparaît. A cet instant, la seule fonctionnalité qui lui est proposée est de sélectionner un terme dans cette liste[23]. Après avoir choisi l'un des termes de la liste, il peut choisir une unité d'apprentissage (UA) que ce terme référence pour en examiner l'aperçu. C'est seulement à ce moment que les deux dernières fonctionnalités sont activées : choisir une page ou retourner au glossaire.

4.3.1.1. Choix d'un terme

Dès son entrée dans le glossaire, l'apprenant pourra choisir un terme dans la liste déroulante affichée sur l'interface (par un simple ou double "clic" de souris, ou par des déplacements dans la liste grâce aux flèches du clavier). Chaque sélection d'un terme fera apparaître tous ses attributs dans les champs appropriés de l'interface. Ces attributs sont : l'abréviation du terme, ses traductions anglaise, néerlandaise, allemande et espagnole, sa définition, une liste de termes apparentés, une liste de termes opposés et une liste d'UA référencées par ce terme.

---

[23] Voir annexe B : "Apprenant : Glossaire" pour un aperçu de cette interface.

Si le glossaire est vide, la liste des termes du glossaire sera également vide.

4.3.1.2. Aperçu d'une unité d'apprentissage

L'apprenant peut visionner un aperçu[24] de l'UA sélectionnée dans la liste des UA référencées. Cet aperçu est composé uniquement du texte des pages de cette UA. Les illustrations, animations et exercices n'y figurent pas. De fait, l'objet de l'aperçu n'est pas de consulter l'UA dans un processus d'apprentissage, mais bien d'obtenir des renseignements complémentaires à la définition d'un terme du glossaire.

4.3.1.3. Choix d'une page

Une UA se compose de plusieurs pages qui doivent être consultées une à une. Ces pages sont numérotées afin de respecter la structure linéaire imposée par l'apprentissage d'une UA. Pour visionner une page particulière, l'apprenant est invité à la choisir dans une liste mise à sa disposition sur l'interface.

4.3.1.4. Retour au glossaire

Quand l'apprenant estime qu'il a obtenu suffisamment d'informations au sujet de cette UA, il peut revenir au glossaire quitté précédemment. Il est alors libre de visionner l'aperçu d'une autre UA référencée par le même terme, sélectionner un autre terme du glossaire ou quitter le glossaire.

4.3.2. Fonctionnalités spécifiques au professeur

Lorsqu'un professeur désire accéder au glossaire du cours sur lequel il travaille, il faut distinguer le cas où ce glossaire est vide et celui où il ne l'est pas. Si le glossaire est vide, le professeur a deux possibilités : charger le glossaire d'un autre cours ou bien créer un nouveau[25] glossaire. Lorsqu'il a fait son choix, le glossaire s'affiche. Ce dernier sera vide si le professeur a décidé de créer un nouveau glossaire. Dans le cas contraire, le glossaire contiendra tous les termes importés. A partir de ce moment,

---

[24] Voir annexe B : "Apprenant : Aperçu UA" pour un aperçu de cette interface.
[25] Voir annexe B : "Professeur : Cas du glossaire vide" pour un aperçu de cette interface.

quel que soit le choix du professeur, celui-ci peut enregistrer des termes, en supprimer ou bien vider tout le contenu du glossaire. Les mêmes possibilités sont proposées au professeur qui accède à un glossaire non vide.

4.3.2.1. Charger un glossaire

Lorsque le glossaire auquel le professeur désire accéder est vide, celui-ci a l'opportunité d'importer celui d'un autre cours. Un tableau contenant un ensemble de cours apparaît alors et l'invite à en sélectionner un[26]. Ce tableau reprend le nom commun et l'identifiant des cours qui sont inscrits dans le système MAI et qui possèdent un glossaire. Quand le professeur a fait son choix, tous les termes du glossaire importé sont chargés[27] et affichés dans l'interface.

4.3.2.2. Créer un glossaire

Le professeur qui décide de créer son propre glossaire voit apparaître à l'écran un glossaire entièrement vide[28]. Il peut alors se mettre à créer un à un les termes de son nouveau glossaire. Pour ce faire, il utilise la fonctionnalité d'enregistrement d'un terme. Si le glossaire n'est pas vide, le professeur a l'occasion de supprimer un ou plusieurs termes ou même tous les termes du glossaire. La suppression d'un terme est effectuée par la fonctionnalité de suppression d'un terme, tandis que l'effacement de tous les termes du glossaire est réalisé par la fonctionnalité de suppression du glossaire.

4.3.2.3. Enregistrer un terme

La création d'un nouveau terme consiste, pour le professeur, à remplir des champs de saisie sur l'interface. Ces derniers correspondent aux différents attributs d'un terme, tel que défini lors de la phase d'analyse : dénomination du nouveau terme, abréviation, traductions, définition, termes apparentés, termes opposés et UA

---

[26] Voir annexe B : "Professeur : Charger un glossaire" pour un aperçu de cette interface.
[27] "chargés" signifie, pour cette fonctionnalité, enregistrés dans la base de données.
[28] Voir annexe B : "Professeur : Créer un nouveau glossaire" pour un aperçu de cette interface.

référencées[29]. Bien entendu, excepté les champs requérant le nom du nouveau terme et sa définition, tous les autres champs ne sont pas obligatoires. Les langues disponibles pour les traductions sont l'anglais, le néerlandais, l'allemand et l'espagnol. Les termes apparentés, de même que les termes opposés, peuvent être des termes provenant du glossaire mais aussi des termes n'y figurant pas. Les UA référencées doivent, quant à elles, nécessairement être choisies parmi celles qui existent et sont, dès lors, proposées dans un tableau indiquant leur nom et leur version. Cette étape d'enregistrement d'un terme se déroule de manière non linéaire. Le professeur peut remplir les champs proposés dans l'ordre qui lui convient. Enfin, un récapitulatif de l'enregistrement permet au professeur de vérifier l'exactitude de toutes les caractéristiques du nouveau terme[30]. Si l'enregistrement se déroule convenablement, un message confirmant la bonne exécution de l'enregistrement du terme dans la base de données s'affiche. Une mise à jour automatique de l'interface intervient alors[31].

4.3.2.4. Supprimer un terme

Un mot sélectionné dans la liste des termes du glossaire peut être supprimé par le professeur. Un message lui demandant la confirmation de la suppression lui permet de ne faire, par mégarde, aucune manipulation erronée. La suppression effectuée, un avertissement informe le professeur de son bon déroulement. L'exécution de cette fonctionnalité s'achève par une mise à jour de l'interface.

4.3.2.5. Supprimer le glossaire

Le professeur a l'opportunité de supprimer tous les termes appartenant au glossaire d'un cours. Un message lui demandant la confirmation de la suppression de tous les termes du glossaire apparaît également dans cette situation. Après approbation, le glossaire est vidé de tout son contenu. Un avertissement informe le professeur de la bonne exécution de la suppression. Il est suivi d'une mise à jour de l'interface.

---

[29] Voir annexe B : "Professeur : Création d'un nouveau terme" pour un aperçu de cette interface.
[30] Voir annexe B : "Professeur : Récapitulatif de l'enregistrement d'un nouveau terme" pour un aperçu de cette interface.
[31] Voir annexe B : "Professeur : Confirmation enregistrement" pour un aperçu de cette interface.

4.3.3. Remarques

Les descriptions des fonctionnalités présentées dans les points précédents sont limitées aux cas où tout se passe bien. Il faut savoir que, dans un souci d'ergonomie maximale de l'application, toutes les mauvaises manipulations de l'utilisateur sont prises en charge et expliquées clairement au moyen de messages d'erreur.

4.4. Spécifications du glossaire

4.4.1. Préambule aux spécifications

Nous avons choisi d'exposer les spécifications du glossaire suivant les techniques des cas d'utilisation et des diagrammes d'interaction proposés par le langage de modélisation orienté utilisateur : UML ("Unified Modelling Language"). Pour mieux comprendre l'utilité de UML, penchons-nous sur la définition que nous donne [FOWLER 00] d'un modèle : "Un modèle est une représentation abstraite d'une spécification, d'un projet ou d'un système. [...] Il est souvent représenté visuellement par un ou plusieurs diagrammes. Il a pour but d'exprimer les caractéristiques essentielles [...] sans donner des détails inutiles." Selon [FOWLER 00], un langage de modélisation est "une manière de représenter les divers modèles produits pendant le processus de développement. La modélisation des besoins a pour objectifs de déterminer les frontières du système, de représenter les interactions entre les utilisateurs et le système, et enfin de structurer les besoins fonctionnels." Par ailleurs, les techniques de modélisation des besoins proposées par UML ont énormément de points communs avec les méthodes de spécification enseignées aux FUNDP, telles que la spécification par pré/post condition, les diagrammes de flux... De plus, nous avons pu observer que UML tend à devenir un standard dans le milieu de la consultance et du développement et ce, à l'échelle internationale. Enfin, l'apprentissage d'un nouveau procédé est toujours une expérience enrichissante dans le domaine de l'informatique.

Un cas d'utilisation, tel qu'il est défini dans le langage UML, est une tâche qui doit être réalisée par un (ou plusieurs) acteur(s), avec le soutien du système. Il permet de capturer les besoins fonctionnels du système (exprimés du point de vue de l'utilisateur) et de favoriser le caractère évolutif de celui-ci. Les diagrammes

représentant les cas d'utilisation servent à modéliser les besoins et à planifier les itérations du développement. Un diagramme de cas d'utilisation se compose d'acteurs et de cas d'utilisation. Chaque cas d'utilisation est symbolisé par un ovale. Les acteurs sont représentés par des personnages dessinés en bâtonnets. Lorsqu'un acteur participe à un cas d'utilisation, une flèche les relie. On peut également intégrer au diagramme la(les) base(s) de données qui intervient(interviennent) dans les cas d'utilisation.

Les diagrammes d'interaction, quant à eux, démontrent comment les objets interagissent pour exécuter une tâche ; ils modélisent le système en action. Ils expliquent de manière détaillée comment un cas d'utilisation se déroule. Il existe deux types de diagrammes d'interaction : les diagrammes de collaboration ("collaboration diagrams") et les diagrammes de séquences ("sequence diagrams"). Ce sont ces derniers que nous avons décidé d'utiliser car ils permettent de représenter graphiquement les interactions dans le temps. Dans ce genre de modélisation, chaque objet intervenant dans un système est représenté par un rectangle étiqueté "NomClasse : NomObjet". Sous chaque objet, une ligne continue représentant sa ligne de vie est tracée. Cette ligne symbolise le temps qui s'écoule (du haut vers le bas). Une flèche reliant les lignes de vie de deux objets évoque un message envoyé par un destinateur vers un destinataire. Lorsque la pointe de la flèche est pleine, le message qu'elle symbolise est un message synchrone. Par contre, une flèche munie d'une pointe évidée représente un appel asynchrone. Un étroit rectangle vertical sur la ligne de vie d'un objet signifie que ce dernier est activé. Une croix mettant un terme à une ligne de vie symbolise la destruction de l'objet correspondant. Enfin, une assertion entre crochets sur un message représente une condition à satisfaire pour pouvoir envoyer ce message.

4.4.2. Cas d'utilisation

4.4.2.1. Représentation des cas d'utilisation

La figure 4.1 fournit une représentation des cas d'utilisation du glossaire du système MAI. On y distingue les deux types d'utilisateurs, ainsi que les accès à la base de données. Ces accès correspondent à des consultations ou à des modifications, selon que la flèche pointe vers le cas d'utilisation ou vers la base de données.

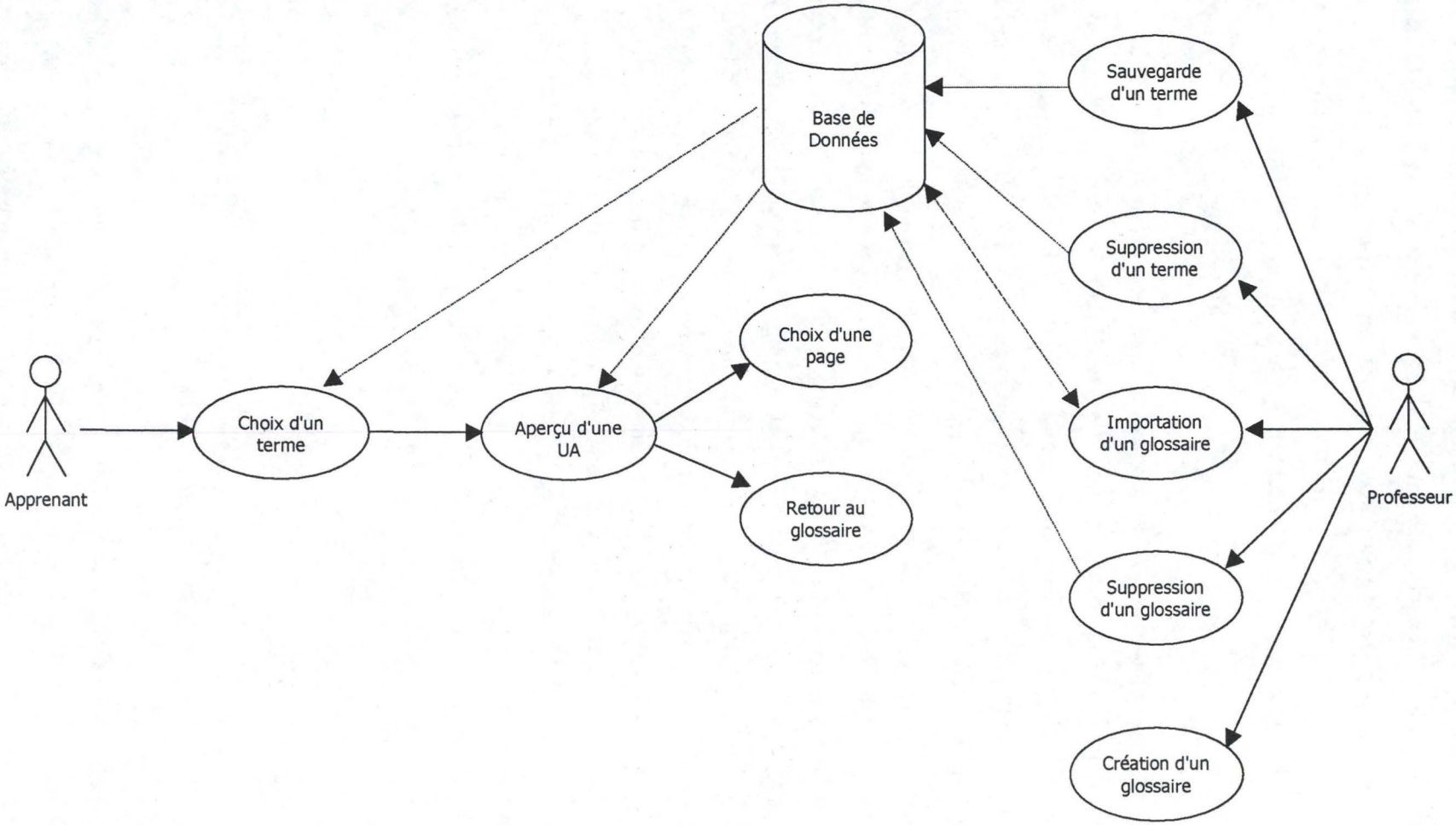


Figure 4.1 : Diagramme des cas d'utilisation du glossaire

4.4.2.2. Description des cas d'utilisation

- **Cas d'utilisation "Choix d'un terme"**

Résumé : Sélection d'un terme afin d'en visionner les attributs

Acteur : Apprenant

Pré-condition(s) :

- L'apprenant a ouvert le glossaire
- Le glossaire n'est pas vide

Description :

Le cas d'utilisation commence quand l'apprenant sélectionne un terme dans la liste des termes qui lui est proposée.

Dans cet ordre :

1) Un accès à la base de données permet de garnir les champs de l'interface au moyen des données correspondant aux attributs du terme sélectionné. Ces attributs sont l'abréviation, les traductions anglaise, néerlandaise, allemande et espagnole, la définition, la liste des termes apparentés, la liste des termes opposés et la liste des UA référencées.
2) S'il le souhaite, l'apprenant peut alors exécuter le cas d'utilisation "Aperçu d'une UA".

Le cas d'utilisation se termine quand l'apprenant quitte le glossaire ou quand il sélectionne un autre terme. Dans ce dernier cas, le cas d'utilisation est ré-initialisé.

Post-condition(s) : Aucune

- **Cas d'utilisation "Aperçu d'une UA"**

Résumé : Demande d'un aperçu de l'UA sélectionnée

Acteur : Apprenant

Pré-condition(s) :

- Etre dans le cas d'utilisation "Choix d'un terme"
- L'apprenant a sélectionné l'une des UA référencées par ce terme
- L'UA sélectionnée contient au moins une page

Description :

Le cas d'utilisation commence quand l'apprenant demande à visionner un aperçu de l'UA qu'il a sélectionnée. Un accès à la base de données permet de charger les pages de cette UA.

Dans cet ordre :

1) Exécution du cas d'utilisation "Choix d'une page" (autant de fois que l'apprenant le souhaite)
2) Exécution du cas d'utilisation "Retour au glossaire" (une seule fois)

Le cas d'utilisation se termine après l'exécution du cas d'utilisation "Retour au glossaire".

Post-condition(s) : Aucune

- **Cas d'utilisation "Choix d'une page"**

Résumé : Sélection d'une page de l'UA afin de la visionner

Acteur : Apprenant

Pré-condition(s) :

- Etre dans le cas d'utilisation "Aperçu d'une UA"

Description :

Le cas d'utilisation commence quand l'apprenant sélectionne l'une des pages de l'UA.

Le contenu de la page choisie s'affiche et l'apprenant peut la visionner.

Le cas d'utilisation se termine quand l'apprenant exécute le cas d'utilisation "Retour au glossaire" ou quand il sélectionne une autre page. Dans ce dernier cas, le cas d'utilisation est ré-initialisé.

Post-condition(s) : Aucune

- **Cas d'utilisation "Retour au glossaire"**

Résumé : Quitter l'aperçu de l'UA afin de retourner au glossaire

Acteur : Apprenant

Pré-condition(s) :

- Etre dans le cas d'utilisation "Aperçu d'une UA"

Description :

Le cas d'utilisation commence quand l'apprenant décide de quitter l'aperçu de l'UA pour retourner au glossaire.

L'apprenant quitte l'aperçu et se retrouve dans le glossaire.

Le cas d'utilisation se termine quand l'apprenant est de nouveau dans le glossaire.

Post-condition(s) : Aucune

- **Cas d'utilisation "Importation d'un glossaire"**

Résumé : Importation du glossaire d'un cours

Acteur : Professeur

Pré-condition(s) :

- Le professeur a tenté d'accéder au glossaire
- Le glossaire est vide

Description :

Le cas d'utilisation commence quand le professeur décide d'importer les termes d'un glossaire.

Dans cet ordre :

1) Accès à la base de données afin de lister les cours possédant un glossaire non vide
2) Choix d'un cours parmi ceux proposés
3) Enregistrement dans la base de données des termes importés du glossaire sélectionné
4) Accès au glossaire contenant les termes importés

Le cas d'utilisation se termine quand le professeur accède au glossaire contenant les termes importés.

Post-condition(s) :

- Le glossaire n'est pas vide

- **Cas d'utilisation "Création d'un glossaire"**

Résumé : Création d'un nouveau glossaire

Acteur : Professeur

Pré-condition(s) :

- Le professeur a tenté d'accéder au glossaire

- Le glossaire est vide

Description :

Le cas d'utilisation commence quand le professeur décide de créer un nouveau glossaire.

Le professeur accède au nouveau glossaire, encore vide. Par la suite, il pourra exécuter le cas d'utilisation "Sauvegarde d'un terme" autant de fois qu'il le désire. De même, il pourra exécuter le cas d'utilisation "Suppression d'un terme" tant que le glossaire n'est pas vide.

Le cas d'utilisation se termine quand le professeur accède au glossaire vide.

Post-condition(s) :

Le glossaire est vide

- **Cas d'utilisation "Suppression d'un glossaire"**

Résumé : Effacement du contenu du glossaire courant

Acteur : Professeur

Pré-condition(s) :

- Le professeur a ouvert le glossaire
- Le glossaire n'est pas vide

Description :

Le cas d'utilisation commence quand le professeur décide d'effacer le contenu du glossaire du cours sur lequel il travaille.

Dans cet ordre :

1) Accès à la base de données pour y effacer tous les termes appartenant au glossaire courant.
2) Mise à jour du glossaire

Le cas d'utilisation se termine quand le glossaire est vide.

Post-condition(s) :

- Le glossaire est vide

- **Cas d'utilisation "Sauvegarde d'un terme"**

Résumé : Enregistrement d'un terme et de ses attributs

Acteur : Professeur

Pré-condition(s) :

- Le professeur a ouvert le glossaire
- Les champs obligatoires ne sont pas vides
- Le terme à enregistrer n'existe pas encore dans le glossaire

Description :

Le cas d'utilisation commence quand le professeur décide d'enregistrer un nouveau terme dans le glossaire du cours sur lequel il travaille.

Dans cet ordre :

1) Affichage d'un récapitulatif de tous les attributs du terme que le professeur désire sauvegarder
2) Enregistrement dans la base de données du terme et de tous ses attributs
3) Mise à jour du glossaire avec le terme créé

Le cas d'utilisation se termine quand le glossaire est mis à jour avec le terme créé.

Post-condition(s) :

- Le glossaire contient le terme créé

- **Cas d'utilisation "Suppression d'un terme"**

Résumé : Effacement d'un terme et de ses attributs

Acteur : Professeur

Pré-condition(s) :

- Le professeur a ouvert le glossaire
- Le glossaire contient le terme à effacer

Description :

Le cas d'utilisation commence quand le professeur décide de supprimer un terme du glossaire du cours sur lequel il travaille.

Dans cet ordre :

1) Effacement dans la base de données du terme et de tous ses attributs
2) Mise à jour du glossaire dont on a supprimé le terme

Le cas d'utilisation se termine quand le glossaire est mis à jour.

Post-condition(s) :

- Le glossaire ne contient plus le terme que le professeur voulait supprimer

4.4.3. Diagrammes d'interaction

4.4.3.1. Diagrammes de séquences

Les trois diagrammes se trouvant aux pages suivantes illustrent les interactions entre les principaux objets du glossaire MAI. Le premier (figure 4.2) est dédié au glossaire de l'apprenant, tandis que les deux autres (figures 4.3 et 4.4) correspondent au glossaire du professeur. Comme nous l'avons déjà maintes fois énoncé, l'application client – serveur se compose d'objets situés soit du côté de l'application cliente, soit du côté de l'application serveur. Cette distinction est illustrée par une ligne épaisse et grisée séparant, sur les figures 4.2, 4.3 et 4.4, le côté client (à gauche) du côté serveur (à droite).

4.4.3.2. Description des diagrammes de séquences

Malgré la différenciation faite entre les scénarios de l'apprenant et du professeur, certains objets sont communs aux deux :

- *gloss*, instance de la classe **MAI_Glossary** ;
- *c*, instance de la classe **ClientLocal** ;
- *sr*, instance de la classe **ServerRemote**.

La classe **MAI_Glossary** est une classe intermédiaire entre le glossaire et l'application cliente qui l'héberge. Elle a pour seul objectif de créer une instance de la classe ClientLocal. Elle lui passe en paramètres la catégorie de l'utilisateur qu'il a entrée lors de son identification ("Professeur" ou "Apprenant"), ainsi que le nom du cours sur lequel il est en train de travailler.

La classe **ClientLocal** a deux buts. Le premier consiste à récupérer, à partir de son pseudonyme, l'instance de l'application serveur distante (*sr*), à laquelle tous les objets du côté client vont pouvoir accéder. Le second objectif de ClientLocal est de créer une instance de la classe Student ou Teacher, selon que la catégorie de l'utilisateur soit "Apprenant" ou "Professeur". Quelle que soit la classe instanciée, elle lui passe en paramètres l'instance de l'application serveur récupérée ainsi que le nom du cours sur lequel l'utilisateur travaille.

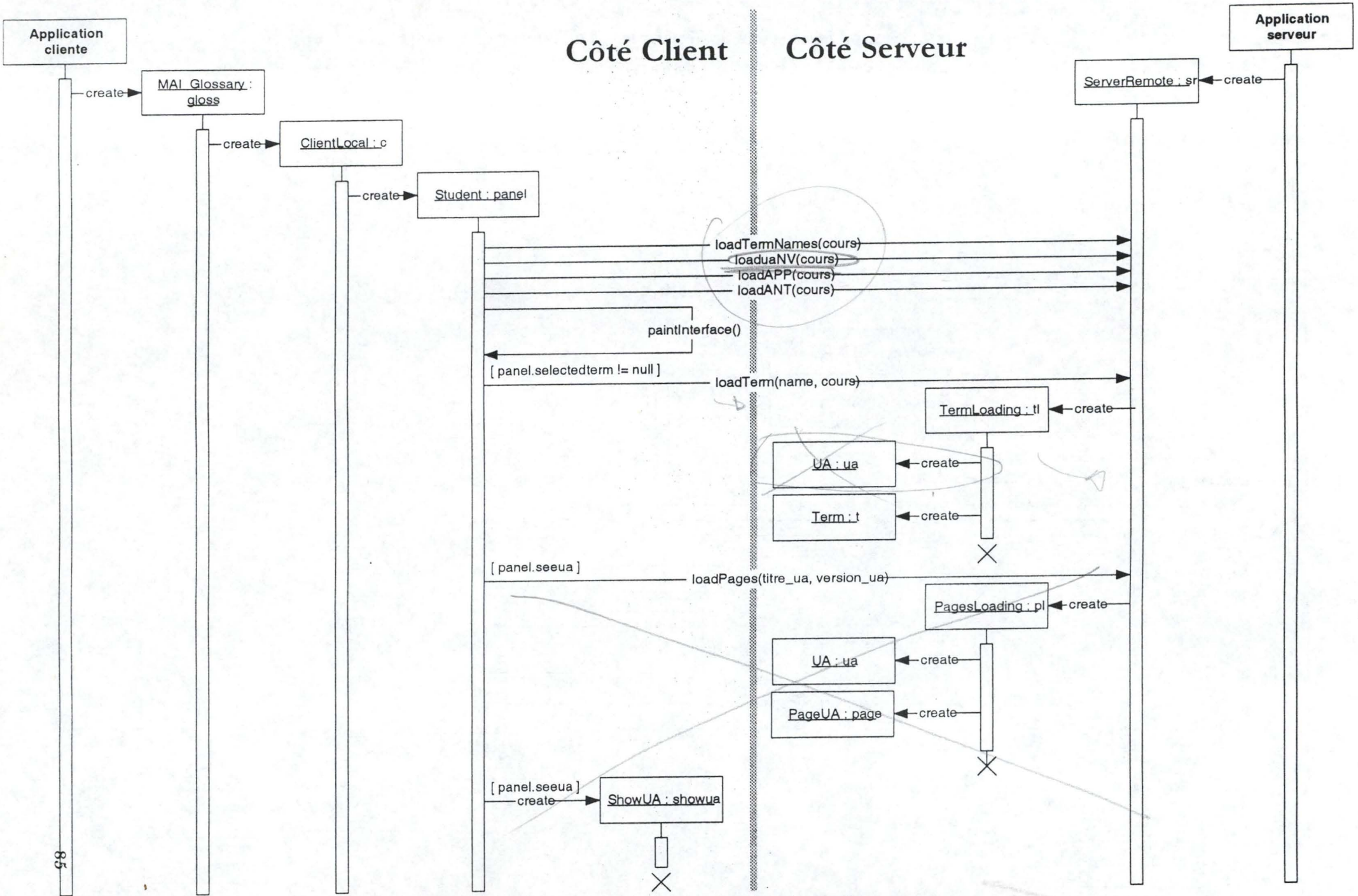


Figure 4.2 : Diagramme d'interaction du glossaire de l'apprenant

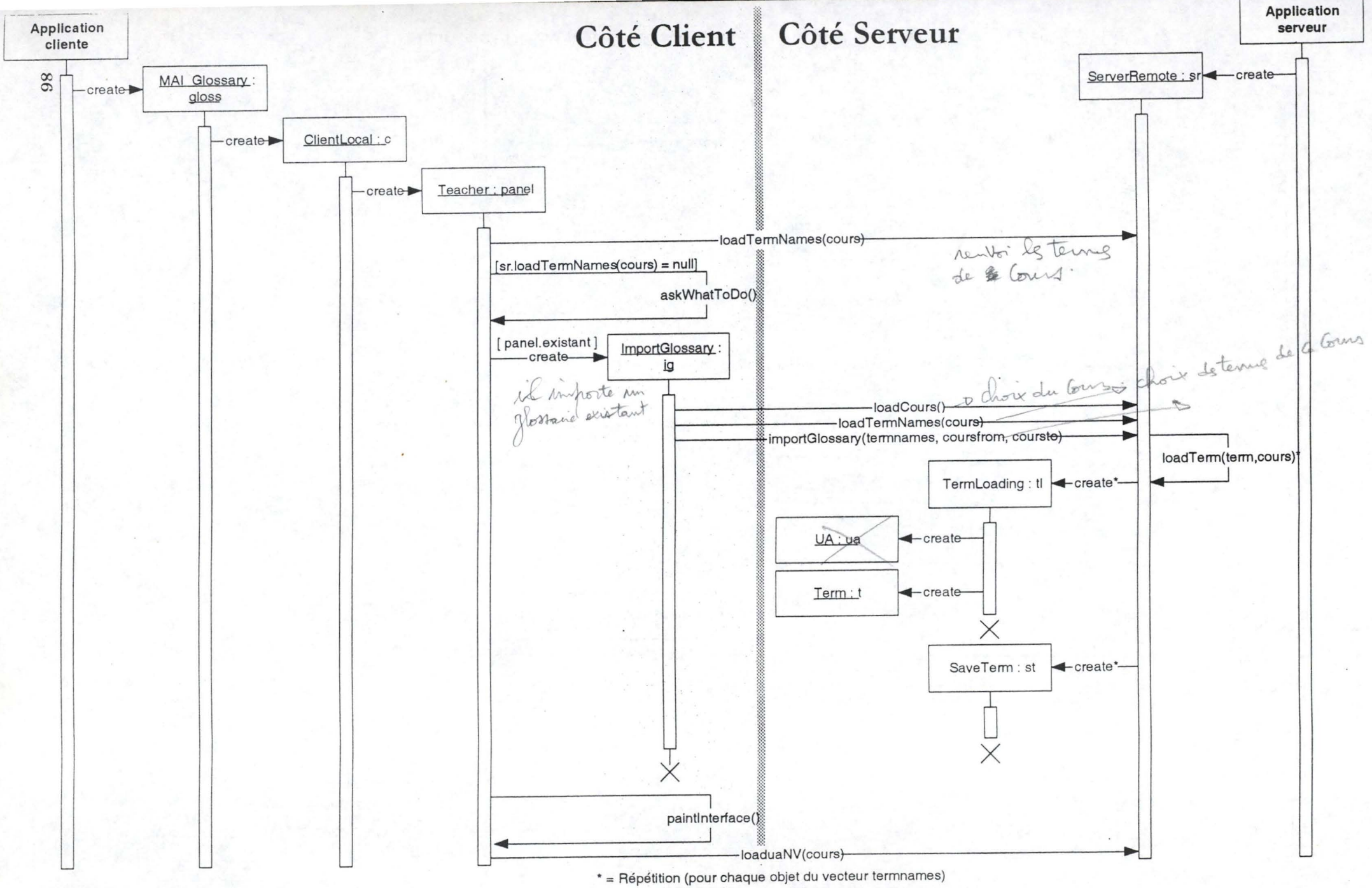


Figure 4.3 : Diagramme d'interaction du glossaire du professeur (1)

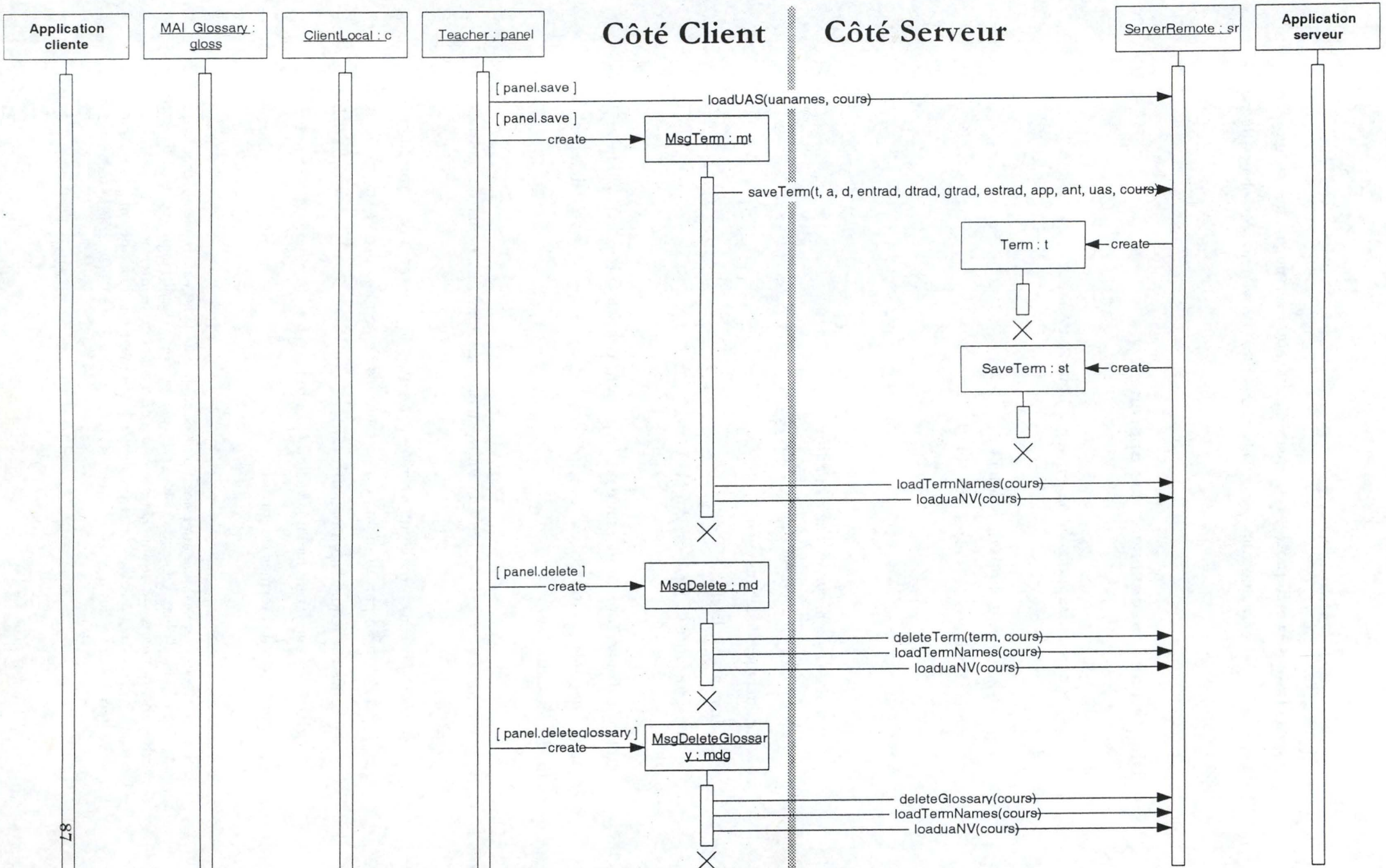


Figure 4.4 : Diagramme d'interaction du glossaire du professeur (2)

La classe **ServerRemote**[32] représente l'application serveur dédiée au glossaire. C'est à son instance *sr* que tous les appels à distance, provenant des objets du côté client, vont s'adresser.

- **Diagramme de séquences de l'apprenant**

Les objets apparaissant du côté client sur la figure 4.2 sont :

- *panel*, instance de la classe **Student** ;
- *showua*, instance de la classe **ShowUA**.

Les objets apparaissant du côté serveur sur la figure 4.2 sont :

- *tl*, instance de la classe **TermLoading** ;
- *pl*, instance de la classe **PagesLoading** ;
- *ua*, instance de la classe **UA** ;
- *t*, instance de la classe **Term** ;
- *page*, instance de la classe **PageUA**.

La phase d'ouverture du glossaire de l'apprenant consiste à garnir le panneau *panel* des noms de tous les termes de ce glossaire. Pour ce faire, *panel* fait appel aux méthodes suivantes de *sr* :

- *loadTermNames(cours)* renvoie un vecteur contenant les noms des termes du glossaire ;
- *loaduaNV(cours)* renvoie un vecteur contenant le nom et la version de chaque UA référencée par au moins l'un des termes de ce glossaire et/ou appartenant au cours sur lequel l'apprenant travaille ;
- *loadAPP(cours)* renvoie un vecteur contenant les noms de tous les termes apparentés des termes du glossaire[33] ;

---

[32] En fait, ServerRemote est une interface qui ne contient que la signature des méthodes qu'elle met à disposition d'objets distants. C'est la classe ServerImpl, implémentant l'interface ServerRemote, qui rend opérationnels les services offerts par cette interface.

[33] Cette méthode permet à l'objet *panel* de calculer la longueur maximale de l'objet de l'interface qui accueillera les noms des termes apparentés. La méthode *loadANT(cours)* a la même utilité.

- *loadANT()* renvoie un vecteur contenant les noms de tous les termes opposés des termes du glossaire.

Quand tous ces éléments ont été chargés, il ne reste plus à *panel* qu'à dessiner l'interface au moyen de la méthode *paintInterface()*.

Lorsque l'apprenant sélectionne un terme dans la liste de termes proposée[34], *panel* fait appel à la méthode distante *loadTerm(name, cours)*. L'objectif final de cette méthode est de créer un objet **Term,** muni de tous ses attributs, et de l'envoyer comme résultat. Pour atteindre cet objectif, la méthode *loadTerm(name, cours)* commence par créer une instance *tl* de la classe **TermLoading** qui va charger tous les attributs du terme dont le nom et le cours auquel il appartient sont *name* et *cours*. Ces attributs sont le nom, l'abréviation, la traduction anglaise, la traduction néerlandaise, la traduction allemande, la traduction espagnole, la définition, la liste des termes apparentés, la liste des termes opposés et la liste des UA. Afin de construire cette dernière liste, il faut créer les objets **UA** un par un à partir des données connues à leur propos. Finalement, l'objet *t* est créé et envoyé à *panel*.

Si l'apprenant décide de visualiser l'aperçu d'une UA référencée par le terme sélectionné[35], *panel* appelle la méthode *loadPages(titre_ua, version_ua)* de *sr* qui a pour objet de renvoyer un vecteur contenant les pages qui composent l'UA sélectionnée. Avant toute chose, la création de *pl*, instance de **PagesLoading**, permet d'aller chercher dans la base de données les attributs de l'UA dont on ne connaît encore que le nom et la version. L'objet **UA** correspondant peut alors être créé. Connaissant le titre et la version de l'UA sélectionnée, on peut également charger toutes ses pages à partir de la base de données. Il suffit ensuite de créer les objets **PageUA** correspondant et de les placer dans le vecteur à renvoyer comme résultat à *panel*. Lorsque *panel* a reçu le vecteur de pages, il peut alors créer *showua*, instance de **ShowUA**, qui affichera une page à la fois (selon le choix de l'apprenant) dans une fenêtre séparée.

---

[34] Cet événement est représenté par la contrainte : [panel.selectedterm != null].
[35] Cet événement est représenté par la contrainte : [panel.seeua].

- **Diagramme de séquences du professeur**

Les objets apparaissant du côté client sur les figures 4.3 et 4.4 sont :

- *panel*, instance de la classe **Teacher** ;
- *ig*, instance de la classe **ImportGlossary** ;
- *mt*, instance de la classe **MsgTerm** ;
- *md*, instance de la classe **MsgDelete** ;
- *mdg*, instance de la classe **MsgDeleteGlossary**.

Les objets apparaissant du côté serveur sur les figures 4.3 et 4.4 sont :

- *tl*, instance de la classe **TermLoading** ;
- *ua*, instance de la classe **UA** ;
- *t*, instance de la classe **Term** ;
- *st*, instance de la classe **SaveTerm**.

La phase d'ouverture du glossaire de l'interface professeur débute par un contrôle de l'état du glossaire du cours courant. Il s'agit, pour l'instance *panel* de la classe **Teacher**, de vérifier si ce glossaire est vide ou pas. Afin d'effectuer ce test, il est nécessaire de faire appel à la méthode *loadTermNames(cours)* de *sr* qui renvoie un vecteur contenant les noms des termes du glossaire appartenant au cours *cours*. Si ce vecteur est vide[36], cela signifie que le glossaire de ce cours est vide. Dans ce cas, la méthode *askWhatToDo()* de *panel* demande au professeur ce qu'il souhaite faire : créer un nouveau glossaire ou importer un glossaire existant. S'il décide d'importer un glossaire existant[37], *panel* crée alors une instance *ig* de la classe **ImportGlossary**. Cette classe a pour objectif ultime d'importer tous les termes du glossaire choisi vers le nouveau glossaire. Avant toute chose, le professeur doit choisir le cours dont il désire importer le glossaire. Afin de lui proposer la liste des cours qui possèdent un glossaire non vide, *ig* appelle la méthode *loadCours()* de *sr*, qui renvoie un vecteur contenant, pour chaque cours, son nom familier et son identifiant. Quand le professeur a choisi un cours parmi ceux proposés, *ig* charge les noms des termes de son glossaire par

[36] Cette situation est représentée par la contrainte : [sr.loadTermNames(cours) = null].
[37] Cet événement est représenté par la contrainte : [panel.existant].

l'invocation de la méthode *loadTermNames(cours)* de *sr*. Maintenant qu'il connaît les noms des termes à importer, ainsi que le cours possédant le glossaire auquel ils appartiennent, *ig* peut appeler la méthode de *sr* appelée *importGlossary(termnames, coursfrom, coursto)*. Cette méthode a pour effet de parcourir le vecteur *termnames* contenant les noms des termes à importer et, pour chaque nom[38], d'appeler la méthode *loadTerm(term, cours)* de *sr*. Comme pour l'apprenant, cette méthode consiste à créer une instance *tl* de la classe **TermLoading** qui va, à son tour, charger les attributs du terme, dont les **UA** référencées. Lorsqu'il possédera tous les attributs du terme, sous leur forme finale, *tl* va créer l'objet *t*, instance de la classe **Term**. L'importation d'un glossaire ne se termine pas par la création de tous les objets Term du glossaire importé. En effet, il faut encore les enregistrer dans la base de données (dans le nouveau glossaire). Cette opération est effectuée par la classe **SaveTerm**.

Quels que soient l'état initial du glossaire (vide ou pas) et le choix du professeur dans le cas du glossaire vide (créer ou importer un glossaire), on aboutit à l'invocation de la méthode *paintInterface()* de *panel* qui a pour effet de dessiner l'interface du glossaire professeur. Cette méthode, lors de son exécution, fait appel à la méthode distante *loaduaNV(cours)* de *sr*. Celle-ci, tout comme dans le glossaire de l'apprenant, renvoie un vecteur contenant le nom et la version de chaque UA appartenant au cours sur lequel le professeur travaille, que cette UA soit référencée ou non par l'un des termes de ce glossaire. Dès cet instant, la liste des termes apparaît sur l'interface. Elle peut se présenter sous différentes formes :

- liste des termes du glossaire initialement non vide ;
- liste des termes importés d'un glossaire existant ;
- liste vide dans le cas de la création d'un nouveau glossaire.

Le professeur peut alors modifier cette liste à loisir en y ajoutant de nouveaux termes ou en en supprimant. Ces opérations sont expliquées dans les paragraphes qui suivent.

Lorsque le professeur manifeste le souhait de sauvegarder un terme et ses attributs[39], il faut récupérer tous les éléments sur l'interface, entrés par le professeur, dans des champs de saisie ou dans des listes. Tous ces attributs sont des chaînes de caractères aisées à récupérer. Seule la liste des noms et versions des UA référencées

---

[38] Cette notion de boucle est symbolisée par une astérisque * sur le diagramme d'interaction. La condition de la boucle est expliquée dans une légende sous le diagramme.

[39] Cet événement est représenté par la contrainte : [panel.save].

doit être transformée en vecteur de véritables objets UA. Pour ce faire, *panel* utilise la méthode *loadUAS(uanames, cours)*, de *sr*, qui, à partir des noms et versions d'UA et du cours actuel, renvoie un vecteur contenant les objets UA correspondants. Par la suite, *panel* va créer une instance *mt* de la classe **MsgTerm**, dont l'objectif est de faire un récapitulatif des attributs du nouveau terme à enregistrer et de le faire valider. Si le professeur le ratifie, *mt* va terminer sa tâche par enregistrer le nouveau terme ainsi que ses attributs dans la base de données et mettre à jour l'interface du glossaire du professeur. Enregistrer le nouveau terme sera exécuté par un appel à la méthode de *sr* appelée *saveTerm(t, a, d, entrad, dtrad, gtrad, estrad, app, ant, uas, cours)*. Effectivement, cette méthode va créer l'objet **Term** correspondant aux données introduites par le professeur, pour ensuite le sauvegarder dans la base de données, grâce à la création de *st*, instance de la classe **SaveTerm**. Mettre à jour l'interface se fera grâce à l'appel aux méthodes *loadTermNames(cours)* et *loaduaNV(cours)* de *sr*.

Dans le cas où le professeur désire supprimer un terme qu'il a sélectionné dans la liste des termes du glossaire[40], *panel* crée *md*, instance de la classe **MsgDelete**. Cette classe sert à effacer un terme de la base de données, de même que tous ses attributs, au moyen de la méthode distante *deleteTerm(term, cours)*. Elle doit ensuite actualiser l'interface par l'invocation des deux mêmes méthodes que précédemment : *loadTermNames(cours)* et *loaduaNV(cours)*.

Quand le professeur décide de supprimer tout le glossaire du cours sur lequel il travaille[41], *panel* crée l'instance *mdg* de la classe **MsgDeleteGlossary**. Cette classe fait appel, tout d'abord, à la méthode de *sr* appelée *deleteGlossary(cours)*. Cette dernière a pour effet d'effacer de la base de données tous les termes du glossaire courant, ainsi que tous leurs attributs. Finalement, *mdg* actualise l'interface du glossaire de façon identique aux deux événements présentés dans les paragraphes précédents, en faisant appel aux méthodes *loadTermNames(cours)* et *loaduaNV(cours)*.

---

[40] Cet événement est représenté par la contrainte : [panel.delete].
[41] Cet événement est représenté par la contrainte : [panel.deleteglossary].

4.5. Conception du glossaire

Nous avons créé 27 classes Java. Ce nombre important de classes résulte d'une volonté de respecter les caractéristiques d'encapsulation et d'héritage de ce langage de programmation. Cette fragmentation maximale permet également une meilleure lisibilité et une ré-utilisabilité plus aisée. Parmi ces classes, certaines se trouvent du côté serveur et d'autres du côté client. Néanmoins, quelques classes se situent de part et d'autre de l'architecture client-serveur. Le lecteur trouvera, en annexe H, le code des classes Java correspondant à l'implémentation de l'outil réalisé dans le cadre de ce mémoire.

4.6. Stockage des données

Cette section vise à décrire la partie de la base de données MAI consacrée au glossaire.

4.6.1. Analyse et modélisation

Dès le début de la phase de conception du glossaire, l'occasion nous a été donnée d'imaginer de nouvelles tables qui devraient venir se greffer sur les tables existantes. Par conséquent, nous avons entrepris une analyse de la base de données MAI afin de concevoir une solution qui soit en harmonie avec elle. Cette base de données a été modélisée par un schéma conceptuel et un schéma logique relationnel. Le premier décrit le modèle Entité-Association, alors que le second décrit les structures d'une base de données selon le modèle d'une famille de SGBD : relationnel, orienté-objet, CODASYL, COBOL... Par conséquent, un schéma conceptuel n'est pas acceptable tel quel par un ordinateur car aucun SGBD n'est basé sur le modèle Entité-Association. Un schéma logique peut, quant à lui, être géré techniquement par un SGBD. Après nous être penchée sur le schéma conceptuel de la base de données MAI, nous avons tiré la conclusion selon laquelle une seule nouvelle table était nécessaire[42].

[42] Le schéma conceptuel décrivant le modèle Entité-Association, il semblerait logique d'utiliser les termes de ce modèle. Ainsi, une "table" correspondrait à un "type d'entités" dans le modèle Entité-Association. Néanmoins, nous gardons la dénomination "table" dans un souci de cohérence avec le reste de ce travail.

Cette table, appelée *Term*, regrouperait tous les termes et leurs attributs, quel que soit leur type. *Term* serait reliée à la table *Cours* existant dans la base de données MAI grâce à un type d'association "un-à-plusieurs" appelé *Appartenance.* De même, un type d'association "plusieurs-à-plusieurs" nommé *Characterization* unirait la table *Term* avec la table *UA* de la base de données MAI, afin de permettre les accès aux UA référencées par chaque terme de *Term.* Nous avons ainsi élaboré un schéma conceptuel (Figure 4.5) de cette nouvelle partie de la base de données, en relation avec les tables *Cours* et *UA.*

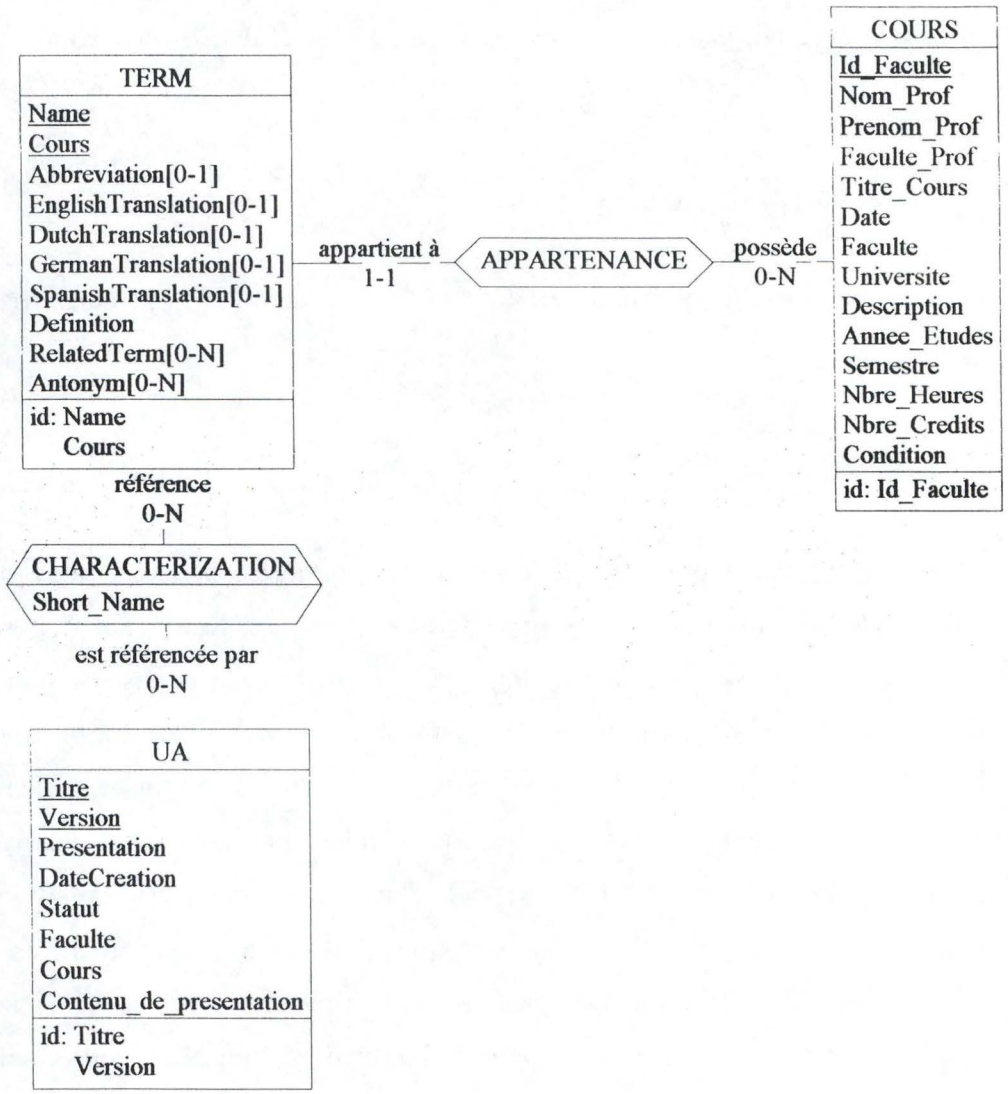


Figure 4.5 : Schéma conceptuel de la partie de la BD consacrée au glossaire

Dans l'étape suivante de notre analyse, nous nous sommes concentrée sur la transformation du schéma conceptuel en un schéma logique implantable sur un ordinateur. Nous avons opté pour un schéma logique relationnel afin de concorder avec la base de données relationnelle du projet MAI. Un schéma logique relationnel est un schéma logique qui ne contient que des structures et des contraintes directement et

Néanmoins, l'outil DB-Main nous a aidée à valider le schéma logique ainsi créé (Figure 4.6). La grande différence entre les deux schémas consiste en l'ajout de trois tables. La première table, intitulée *Characterization*, résulte de l'application exacte de la règle 5. Le lecteur remarquera que l'attribut du type d'associations devient simplement une colonne de la nouvelle table. Les deux autres tables, *Relatedterm* et *Antonym*, ne découlent pas directement des règles présentées à la page précédente. De fait, elles correspondent à une première transformation du schéma conceptuel dans lequel les attributs multivalués (par opposition aux attributs monovalués et atomiques) ont été traduits en nouveaux types d'entités. Chaque type d'entités est relié au type d'entités initial par un type d'associations "un-à-plusieurs". Il a alors suffit de traduire ces nouveaux types d'entités et d'associations en appliquant les règles 1, 2 et 3. Les tables *Relatedterm* et *Antonym* rassemblent respectivement le nom des termes apparentés et le nom des termes opposés à chaque terme de la table *Term*.

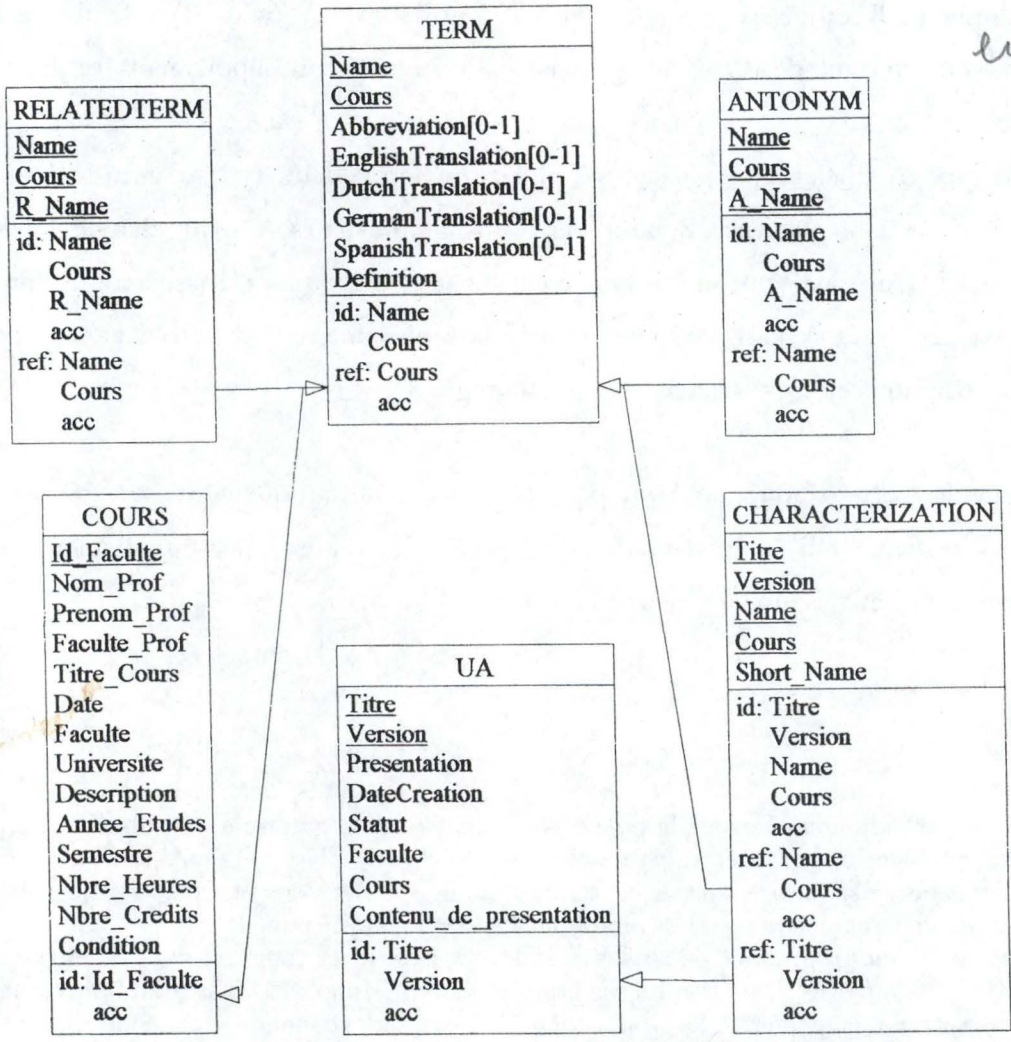


Figure 4.6 : Schéma logique relationnel de la partie de la BD MAI consacrée au glossaire

explicitement exprimables dans le modèle de SGBD. Ainsi, nous avons synthétisé les règles de base de la transformation d'un schéma conceptuel vers un schéma logique relationnel proposées par [HAINAUT 94] :

1) Un type d'entités est traduit par une table.
2) Un attribut est traduit par une colonne.
3) Un type d'associations "un-à-plusieurs" entre deux types d'entités A et B, tel qu'il existe plusieurs B pour un A et un seul A pour chaque B, se traduit par une clé étrangère de B vers A. Cette clé étrangère est une nouvelle colonne de B et correspond à l'identifiant de A.
4) Un type d'associations "un-à-un" entre deux types d'entités A et B, tel qu'il existe un seul B pour chaque A et un seul A pour chaque B, se traduit par une clé étrangère de B vers A (ou de A vers B[43]). Cette clé étrangère est une nouvelle colonne de B (ou de A) et correspond à l'identifiant de A (ou de B). De plus, cette nouvelle colonne de B (ou de A) constitue un identifiant supplémentaire pour sa table.
5) Un type d'associations "plusieurs-à-plusieurs" entre deux types d'entités A et B, tel qu'il existe plusieurs B pour chaque A et plusieurs A pour chaque B, doit d'abord être transformé en un type d'entités C et deux types d'associations "un-à-plusieurs" vers A et B. On poursuit en transformant C en table et les types d'associations en clés étrangères, selon la règle 3.

Même si cette traduction peut être effectuée automatiquement par des outils existants tels que DB-Main[44], nous avons préféré la réaliser personnellement, afin d'exercer nos connaissances en ce domaine[45].

---

[43] Le choix est indifférent lorsque le type d'associations est obligatoire ou facultatif pour A et pour B. Par contre, la clé étrangère sera obligatoirement placée chez A lorsque le type d'associations est obligatoire pour A et facultatif pour B. Inversement, elle sera ajoutée à la table B si le type d'associations est facultatif pour A et obligatoire pour B.

[44] DB-Main est un programme de recherches, développement et transfert de technologie mis sur pied, en 1993, par le "Database Engineering Research Group" de l'Institut d'Informatique. Il est consacré à l'ingénierie des applications de bases de données et a pour objectif le développement de méthodes et d'outils pour la conception, la rétro-ingénierie, la maintenance et l'évolution des bases de données et de leurs applications.

[45] Ces connaissances proviennent de cours dispensés par le professeur J.-L. HAINAUT à l'Institut d'Informatique des FUNDP.

Pendant la dernière phase, nous avons réalisé le script SQL de création de la base de données du glossaire, que nous avons par la suite validé au moyen de l'outil DB-Main. Le lecteur trouvera ce script SQL dans l'annexe G.

Les tables du glossaire, qui sont détaillées dans le paragraphe suivant, ont été intégrées avec succès dans la base de données MAI. Les tables *UA* et *Cours* de la base de données MAI, auxquelles sont reliées les tables du glossaire, ont déjà été décrites dans le chapitre 3, au paragraphe 3.3.6.

IPIX Virtual Tour Vieilles 23

4.6.2. Description des tables

a) Term

Le terme est l'élément principal du glossaire. Il est identifié par sa dénomination et le cours pour lequel il a été créé. La présence du nom du cours est primordiale dans cette table car il représente la clé de décomposition de l'ensemble des termes de la table *Term*. En effet, l'application dont il est question dans ce quatrième chapitre présente à l'utilisateur le glossaire (ensemble des termes) qui concerne le cours sur lequel cet utilisateur est en train de travailler. Dès lors, le nom du cours constitue une clé étrangère vers la table *Cours* de la base de données MAI. Un terme possède obligatoirement une définition. Il peut avoir comme caractéristiques une abréviation, des traductions anglaise, néerlandaise, allemande et espagnole et une illustration. En plus de ces attributs, il peut référencer une liste de termes apparentés, une liste de termes opposés et une liste de maximum quatre UA.

Chaque **term** possède les attributs suivants :
(Les attributs soulignés forment l'identifiant primaire.)

Name
Cours
Abbreviation
EnglishTranslation
DutchTranslation
GermanTranslation
SpanishTranslation
Definition

b) RelatedTerm

Chaque terme apparenté de la table *RelatedTerm* est identifié par le nom du cours, par la dénomination du terme auquel il est apparenté et par son propre nom. Les deux premiers attributs constituent une clé étrangère vers la table *Term*. Un terme apparenté ne doit pas nécessairement faire partie d'un glossaire, c'est-à-dire appartenir à l'ensemble des termes enregistrés dans la table *Term*. En effet, la situation inverse provoquerait des blocages en ce sens que, lorsque tous les glossaires sont vides, il serait impossible d'associer un terme apparenté à un nouveau terme.

Chaque **relatedterm** possède les attributs suivants :
(Les attributs soulignés forment l'identifiant primaire.)

Name
Cours
R_Name

c) Antonym

Chaque terme opposé de la table *Antonym* est identifié par le nom du cours, par la dénomination du terme duquel il est un antonyme et par son propre nom. Les deux premiers attributs constituent une clé étrangère vers la table *Term*. Pour la même raison que celle énoncée dans la description de la table *RelatedTerm*, un terme opposé ne doit pas nécessairement appartenir à un glossaire.

Chaque **antonym** possède les attributs suivants :
(Les attributs soulignés forment l'identifiant primaire.)

Name
Cours
A_Name

d) Characterization

Chaque enregistrement de la table *Characterization* est identifié par le nom du cours et la dénomination d'un terme de la table *Term*, ainsi que par le titre et la version de l'UA référencée par ce dernier. Les deux premiers attributs constituent une clé étrangère vers la table *Term*. Les deux derniers constituent une clé étrangère vers la table *UA*. Un attribut supplémentaire donne le nom court de l'UA qui sera utilisé dans l'interface du glossaire au cas où le titre de celle-ci serait trop long.

Chaque **characterization** possède les attributs suivants :

(Les attributs soulignés forment l'identifiant primaire.)

Name
Cours
Titre
Version
Short_Name

Conclusion

L'objectif de ce mémoire était le développement d'une application correspondant à la partie glossaire d'un système d'apprentissage multimédia, objet du projet MAI des FUNDP. L'outil créé devait, d'une part, permettre à un apprenant de consulter la définition ainsi que d'autres caractéristiques d'un terme inconnu ou compliqué. D'autre part, cet outil avait pour but d'aider un professeur à créer un glossaire, en importer un ou en modifier un en effaçant certains de ses termes et en y insérant de nouveaux. Pour ce faire, la base de données du projet MAI devait être étendue aux nouvelles informations nécessaires au glossaire.

Travail réalisé

Le travail réalisé dans le cadre de ce mémoire a comporté les étapes suivantes :

- une étape de prise de connaissance de l'équipe et du projet MAI ;
- une étape de familiarisation avec le développement d'applications Web et avec le langage de programmation Java, grâce à un stage en Grande-Bretagne ;
- une étape de réflexion sur ce qu'est un glossaire, au moyen de recherches approfondies en bibliothèque ;
- une étape d'analyse des besoins mise en œuvre par la récolte des exigences des membres du projet MAI et par un examen minutieux de deux cents glossaires "on-line" ;
- une étape de spécification des besoins au moyen du langage de modélisation UML ;
- une étape de validation de la maquette réalisée et des fonctionnalités proposées auprès des membres de l'équipe MAI ;

- une étape de création de la partie de la BD MAI consacrée au glossaire ;

- une étape de programmation en Java de l'outil demandé ;

- une étape d'intégration du programme réalisé dans le système MAI (en cours de développement).

Perspectives

Dans son état actuel, l'application créée est tout à fait fonctionnelle. Néanmoins, elle pourrait être améliorée par l'ajout de certaines fonctionnalités.

Il avait été envisagé de proposer, à l'utilisateur du système MAI, les deux fonctionnalités suivantes : l'accès au glossaire à partir d'une unité d'apprentissage et l'affichage, dans un "ToolTip", de la définition d'un terme d'une unité d'apprentissage. Des méthodes situées sur le serveur ont été mises à disposition de l'équipe MAI dans le but de faciliter au maximum l'implémentation de ces deux fonctionnalités. Cependant, ces dernières n'ont pu être développées en raison de l'état d'avancement du développement du système auteur concernant le traitement et l'affichage des unités d'apprentissage.

Ensuite, nous avons réfléchi à l'opportunité de permettre au créateur d'un nouveau terme dans un glossaire d'insérer des illustrations dans la définition de ce terme. En effet, les images et animations constituent une des caractéristiques majeures des systèmes multimédias. Nous nous sommes demandé quelle structure utiliser pour traiter simultanément du texte et des images, sous quelle forme les stocker dans la base de données et, surtout, comment les extraire de cette base de données pour les afficher. La même difficulté était observée par un informaticien de l'équipe MAI lors du développement des procédures d'aide à la création et d'affichage des unités d'apprentissage. Au moment où nous écrivons ces lignes, cette question est encore en suspens.

Une troisième amélioration de l'outil pourrait consister en l'intervention du son, particularité importante du domaine du multimédia, dans la définition et/ou dans

les autres attributs d'un terme du glossaire. Il pourrait être possible d'écouter, par exemple, la prononciation d'un terme. Par ailleurs, étendre cette sonorisation à tous les attributs d'un terme pourrait rendre le glossaire accessible aux malvoyants.

En outre, l'application élaborée pourrait être améliorée par quelques petits changements tels que :

- proposer à l'auteur d'un glossaire de choisir lui-même dans quelles langues les termes du glossaire posséderont une traduction ;
- présenter plusieurs traductions d'un terme dans une même langue ;
- afficher l'entièreté de l'UA (texte, illustrations, animations...) lorsque l'apprenant demande à voir un aperçu.

Il s'agirait également d'uniformiser les interfaces du glossaire et du reste du système MAI et de rechercher une ergonomie maximale et homogène. Ces améliorations sont primordiales pour susciter l'intérêt des futurs utilisateurs.

En définitive, il nous semble qu'une phase de validation, mise en œuvre par des batteries de tests et par l'intégration du volet biologie, est essentielle si l'on veut offrir aux futurs utilisateurs un système d'apprentissage agréable et facile à employer. Cette phase serait, dès lors, l'occasion de rassembler une dernière fois des chercheurs d'horizons différents – biologistes et informaticiens -, autour d'un projet commun : aider tout un chacun à avancer de quelques pas dans le vaste champ de la connaissance.

Bibliographie

• Références bibliographiques

[ACREMANN-DUPIN-MOUJEARD 99] **ACREMANN D., DUPIN S. & MOUJEARD G.**, *"Le programmeur : Borland® JBuilder 3"*, Paris, CampusPress France, 1999.

[ALLEN 94] **ALLEN R.**, *"Chambers Encyclopedic English Dictionary"*, Chambers, 1994.

[BARNHART-BARNHART 92] **BARNHART C. L. & BARNHART R. K.**, *"The World Book Dictionary"*, t. I et II, "*Volume one A-K*" et "*Volume two L-Z*", Chicago – London –Sydney - Toronto, World Book, Inc. a Scott Fetzer Company, 1992.

[BODART-PIGNEUR 94] **BODART F. & PIGNEUR Y.**, *"Conception assistée des systèmes d'information* Méthode – Modèles - Outils", Paris – Milan - Barcelone, MASSON, 1994, MIPS.

[CAMBRIDGE 95] **PROCTER P.**, *"Cambridge International Dictionary of English"*, Cambridge, Cambridge University Press, 1995.

[FOWLER 00] **FOWLER M.**, *"UML Distilled : A Brief Guide to the Standard Object Modeling Language"*, Second Edition, Reading – Harlow – Menlo Park – Berkeley – Don Mills – Sydney – Bonn – Amsterdam – Tokyo – Mexico City, ADDISON – WESLEY, 2000.

[HAINAUT 94] **HAINAUT J.-L.**, *"Bases de données et modèles de calcul : Outils et méthodes pour l'utilisateur"*, Paris, InterEditions, 1994.

[HAWKINS-ALLEN 91] **HAWKINS J. M. & ALLEN R.**, "*The Oxford Encyclopedic English Dictionary*", Oxford, Clarendon Press, 1991.

[LAROUSSE 73] **LAROUSSE P.**, *"Grand Larousse de la langue française en six volumes"*, Paris, Librairie Larousse, 1973.

[LAROUSSE 87] **LAROUSSE P.**, *"petit Larousse illustré"*, Paris, Larousse, 1987.

[LAROUSSE 89] **LAROUSSE P.**, *"Grand Larousse Universel"*, Paris, Larousse, 1989.

[LEMAY-CADENHEAD 00] **LEMAY L. & CADENHEAD R.**, *"Le programmeur : Java 2"*, Paris, CampusPress France, 2000.

[MILAN 00] **MILAN**, *"Droit au but : Découvrir et comprendre HTML"*, Paris, Osman Eyrolles Multimedia, 2000.

[NORTON-ESPOSITO 95] **NORTON P. B. & ESPOSITO J. J.** , "*The New Encyclopaedia Britannica*", 15th Edition, Chicago, Encyclopaedia Britannica Inc., 1995.

[ROBERT 95] **ROBERT P.**, *"Le nouveau petit Robert"*, Paris, Dictionnaires Le Robert, 1995.

[SIMPSON-WEINER 89] **SIMPSON J.A. & WEINER E.S.C.**, "*The Oxford English Dictionary*", Second Edition, Oxford, Clarendon Press, 1989.

[SKEAT 10] **SKEAT W. W.**, *"An Etymological Dictionary of the English Language"*, New edition revised and enlarged, Oxford, Clarendon Press, 1910.

[VANDERDONCKT 95] **VANDERDONCKT J.**, *"Description d'objets interactifs abstraits"*, s. l., s. é., 1995.

[VOSS 98] **VOSS A.**, " *Dictionnaire de l'Informatique et de l'Internet* ", Paris, Micro Application, 1998.

- **Références sur Internet**

http://encarta.msn.com/find/Concise.asp?ti=0B3AB000

Glossaire encyclopédique en ligne de Microsoft

http://looksmart.altavista.com/cgi-bin/query?pg=dir&tp=Library/Sciences/Earth.26_Environment/Marine_Science/Reference_Shelf/Glossary&crid=589297

Glossaire proposé par le moteur de recherches "Altavista"

http://www.nyse.com/help/glossary.html

Glossaire de la bourse de New York

http://www.investorwords.com/

Glossaire des termes employés par les investisseurs

http://www.nasd.com/glossary/n.html

Glossaire des termes du Nasdaq

http://java.sun.com/docs/glossary.html

Glossaire des termes utilisés en Java

http://isp.webopedia.com/

Glossaire des termes utilisés par les "Internet Service Providers"

http://www.realtor.com/Glossary/default.asp

Glossaire des termes utilisés par une agence immobilière

http://www.csgnetwork.com/glossary.html

Glossaire de l'industrie informatique, téléphonique et électronique

http://www.moverquotes.com/glossary.html

Glossaire du déménagement

http://shoga.wwa.com/~rgs/glossary.html

Glossaire des termes de poésie

http://allserv.rug.ac.be/~rvdstich/eugloss/welcome.html

Glossaire plurilingue médical

http://www.weihenstephan.de/~schlind/genglos.html

Glossaire de génétique

http://www.trp.dundee.ac.uk/research/glossary/glossary.html

Glossaire de la conservation des constructions urbaines

http://www.mhhe.com/biosci/abio/glossary.mhtml

Glossaire d'anatomie

http://www.zoology.ubc.ca/courses/bio205/

Glossaire de zoologie

http://www.isostar.tm.fr/nutrition/glossaire/index.html

Glossaire du Sport et de la Nutrition

http://www.europal.net/Fr/Infos/I/GlosR/dictionnaire/francais/elements/charge.htm

Glossaire d'une entreprise de construction de palettes

http://www.it-seine.com/glossaire/

Glossaire des systèmes d'information

http://www.anapath.necker.fr/enseign/glossaireAP/GlossaireApHomePage

Glossaire d'anatomie pathologique

ANNEXES

Annexe A : Glossaires électroniques

Annexe B : Interfaces du glossaire

Annexe C : Glossaire

Annexe D : Exemple d'une UA réalisée par l'équipe de biologistes

Annexe E : Schéma conceptuel de la BD du système MAI

Annexe F : Schéma logique relationnel de la BD du système MAI

Annexe G : Script SQL de création de la partie de la BD du système MAI consacrée au glossaire

Annexe H : Code Java du glossaire du système MAI

ANNEXE A

Glossaires électroniques

Glossaire 2

Tide

Outline | Click to view outline and jump to a section.

Click to view outline and jump to a section.
I. Introduction
II. Lunar Tides
III. Solar Tides
IV. Tidal Currents and Waves
V. Tidal Energy

I. Introd

Print sectio

Tide, per luding those of open sea, gulfs, and bays, resulting from the gravitational attraction of the moon and the sun upon the water and upon the earth itself. *See* Gravitation.

Article Topics

I. INTRODUCTION
II. LUNAR TIDES
III. SOLAR TIDES
IV. TIDAL CURRENTS AND WAVES
V. TIDAL ENERGY

1 media item

Related Items

EDUCATIONAL PRODUCTS
from SmarterKids.com

News and Updates

PERIODICALS
from Electric Library
free registration required

NEWS
from MSNBC

II. Lunar Tides

Print section

The moon, being much nearer to the earth than the sun, is the principal cause of tides. Because the force of gravity decreases with distance, the moon exerts a stronger gravitational pull on the side of the earth that is closer to it and a weaker pull on the side farther from it. The earth does not respond to this variation in strength because the planet is rigid—instead, it moves in response to the average of the moon's gravitational attraction. The world's oceans, however, are liquid and can flow in response to the variation in the moon's pull. On the side of the earth facing the moon, the moon's stronger pull makes water flow toward it, causing a dome of water to rise on the earth's surface directly below the moon. On the side of the earth facing away from the moon, the moon's pull on the oceans is weakest. The water's inertia, or its tendency to keep traveling in the same direction, makes it want to fly off the earth instead of rotate with the planet. The moon's weaker pull does not compensate as much for the water's inertia on the far side, so another dome of water rises on this side of the earth. The dome of water directly beneath the moon is called direct tide, and the dome of water on the opposite side of the earth is called opposite tide.

Glossaire proposé par le moteur de recherches "Altavista"

http://looksmart.altavista.com/cgi-bin/query?pg=dir&tp=Library/Sciences/Earth.26_Environment/Marine_Science/Reference_Shelf/Glossary&crid=589297

Glossaire 1

Find in this article
Print article

Omnivore

Omnivore, an animal that eats both animal flesh and vegetable matter. The term *omnivore* indicates similarities in the behavior and physiology of many unrelated animals; for example, many small birds and mammals are omnivorous.

Because of their wide food preferences, omnivores are usually less specialized in their food gathering habits than either carnivores or herbivores. Depending on food supply, the spotted skunk feeds mainly on mice and rats during the winter, and on seeds and insects in the summer months. Similarly the mockingbird varies its diet between berries and insects, depending on the time of year. Many animals that are described as being carnivorous are actually omnivorous; the grizzly bear supplements its diet of flesh with grasses, herbs, nuts, and berries, and the red fox feeds on fruits and berries as well as animal flesh. Omnivores can be both primary and secondary consumers within the food web.

The digestive systems of omnivores are adapted to greater variety within the diet. The teeth of omnivorous mammals possess special features common to the teeth of both herbivores and carnivores. Human beings are an example; their teeth can tear off flesh and grind up tough plant material.

Glossaire encyclopédique en ligne de Microsoft

http://encarta.msn.com/find/Concise.asp?ti=0B3AB000

Glossaire 4

Home | Search | Help | Suggest a Word | Bookmark | Link | NetWords
License InvestorWords | Advertise | Investorville | InvestorGuide.com

With over 5,000 definitions and 15,000 links between related terms, InvestorWords is the most comprehensive financial glossary you'll find anywhere, online or off. Just click on one of the links below to begin.

Aa-	Ad-	Al-	Ap-	At-	Ba-	Bd-	Bo-	Br-	Caa-	Cas-	Cb-	Ci-	Com
Con	Coo-	Cp-	Da-	Ded-	Di	Dj-	Ea-	En-	Ex	Fa-	Fi-	Fn-	Ga-
Gr-	H	Ia-	Ine-	Int	Inv-	J	K	La-	Li-	Maa-	Mas-	Mo-	Na-
Nf-	Ob-	Op	Or-	Pa	Pe-	Po-	Pri	Pro-	Q	Ra-	Red-	Rep-	Rh-
Sa-	See-	She-	Sta-	Str-	Ta-	Te-	Tr-	U	V	W	X	Y	Z-#

Just click here and a new window will open with our other site, InvestorGuide.com, a comprehensive guide to all the best online investing sites. This window will allow you to look up any unfamiliar words if you come across any.

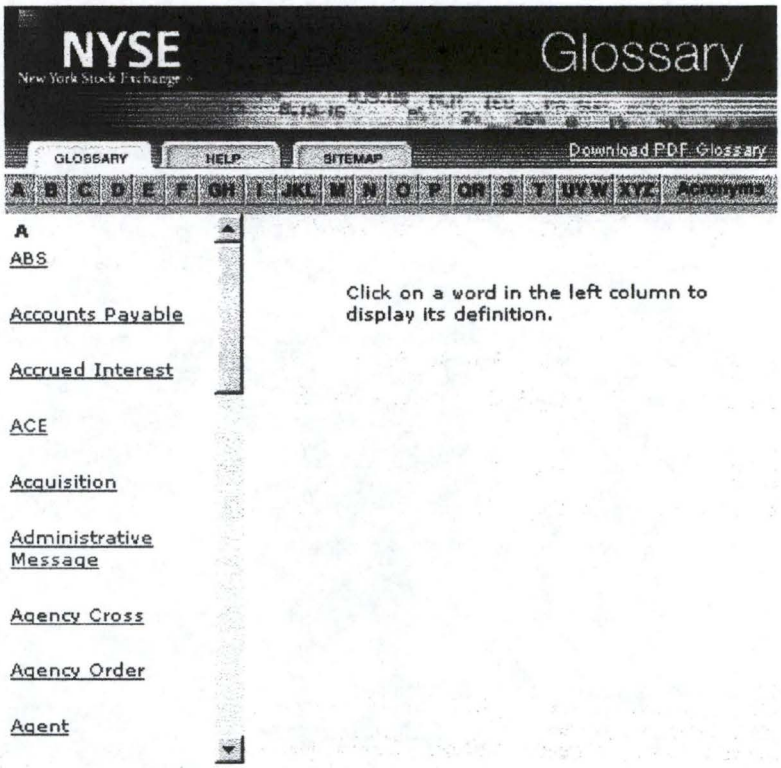
Recent News:
(09/26/00) InvestorWords.com selected as Money magazine's "The Best of the Web"
(05/17/00) InvestorWords.com reaches 50 Licensee Milestone
(02/09/00) InvestorGuide.com Signs InvestorWords Licensing Agreement With E*Trade for Investor Education

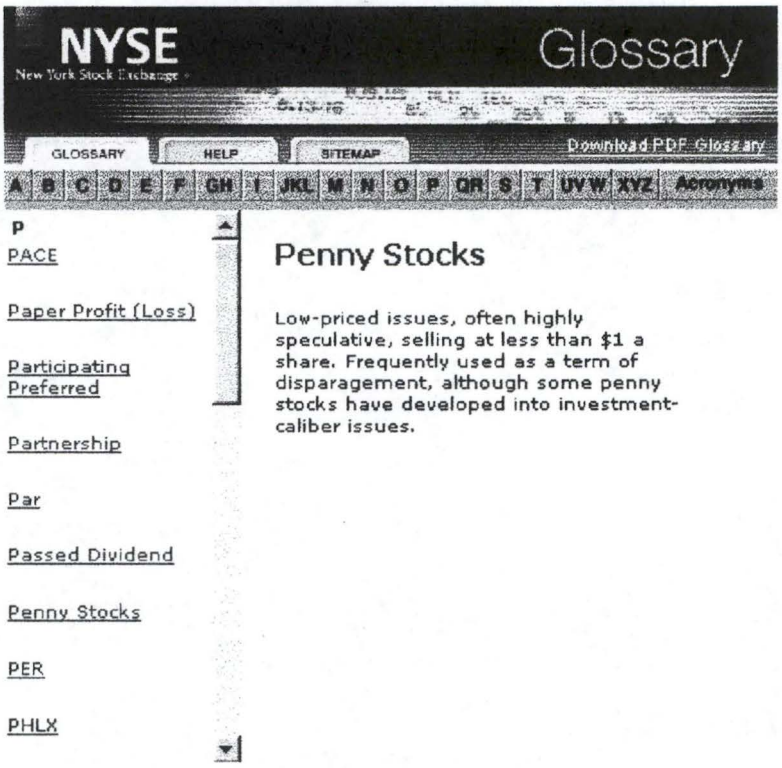
Webmasters: If you operate a high-traffic financial web site and would like to have this glossary for your visitors through a licensing agreement, check out our licensing program.

Glossaire des termes employés par les investisseurs

http://www.investorwords.com/

Glossaire 3

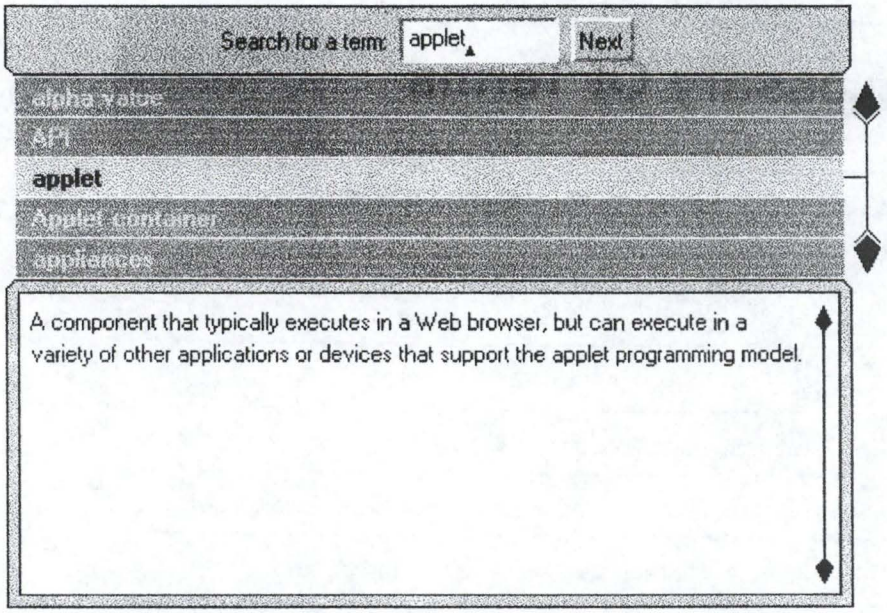




Glossaire de la bourse de New York

http://www.nyse.com/help/glossary.html

Glossaire 6



Glossaire des termes utilisés en Java

http://java.sun.com/docs/glossary.html

A B C D E F G H I L M N O P Q R S T U V W

N

N*PROVE[SM]	See NAqcess[SM]
NAC	See National Adjudicatory Council
NASAA	See North American Securities Administrators Association, Inc.
NASD	See National Association of Securities Dealers, Inc.
NASD Information Request Form (NIRF)	Allows members of the public to obtain certain types of disciplinary and registration information regarding member firms and associated persons.
NASDR[SM]	See NASD Regulation, Inc.

Glossaire des termes du Nasdaq

http://www.nasd.com/glossary/n.html

Glossaire 8

GLOSSARY

| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |

W

what-if analysis
An affordability analysis that is based on a what-if scenario. A what-if analysis is useful if you do not have complete data or if you want to explore the effect of various changes to your income, liabilities, or available funds or to the qualifying ratios or down payment expenses that are used in the analysis.

what-if scenario
A change in the amounts that is used as the basis of an affordability analysis. A what-if scenario can include changes to monthly income, debts, or down payment funds or to the qualifying ratios or down payment expenses that are used in the analysis. You can use a what-if scenario to explore different ways to improve your ability to afford a house.

wraparound mortgage
A mortgage that includes the remaining balance on an existing first mortgage plus an additional amount requested by the mortgagor. Full payments on both mortgages are made to the wraparound mortgagee, who then forwards the payments on the first mortgage to the first mortgagee.

| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |

Glossaire des termes utilisés par une agence immobilière

http://www.realtor.com/Glossary/default.asp

Glossaire 7

ISP GLOSSARY

Your source for the most up-to-date terms, definitions, and acronyms for and about internet service providers.

Search for an ISP term

Find

Top 10 ISP Terms

- Cable Modem
- DSL
- ISDN
- ISP
- IAP
- Modem
- Online Service
- T-1
- T-3
- VPN

ISP GLOSSARY

DVD-ROM

A new type of read-only compact disc that can hold a minimum of 4.7GB (gigabytes), enough for a full-length movie.

The DVD-ROM specification supports disks with capacities of from 4.7GB to 17GB and access rates of 600 KBps to 1.3 MBps. One of the best features of DVD-ROM drives is that they are backward-compatible with CD-ROMs. This means that DVD-ROM players can play old CD-ROMs, CD-I disks, and video CDs, as well as new DVD-ROMs. Newer DVD players can also read CD-R disks.

DVD-ROMs use MPEG-2 to compress video data.

Related Terms

- Divx
- DVD
- DVD-RAM
- FMD-ROM
- MPEG

More Information

Dolby Laboratories home page
Offers links to company information, news and products, as well as technical information about Dolby Digital, DVD, and Dolby Surround and multimedia.
Updated on Aug 5, 1998

DVD - the new optical disk format
Describes the DVD format, capacity, immediate future, cautionary notes, and links to DVD related Web resources.
Updated on Oct 17, 1997

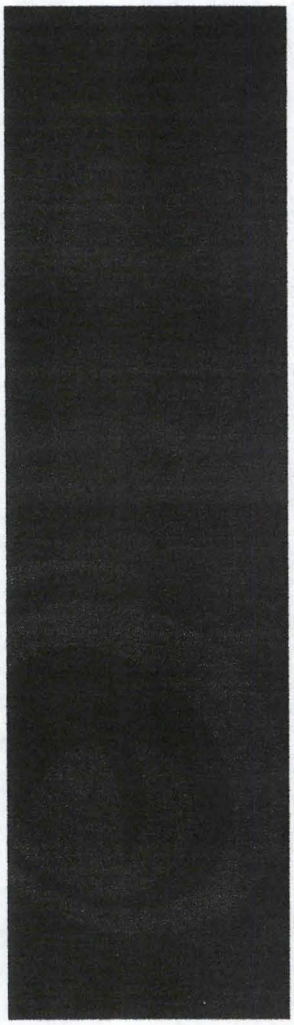
DVD Express
Contains online ordering information for a large selection of DVD titles. Browse film categories or search for a specific title for film details, cover art, trailers and talent listings.
Updated on Aug 5, 1998

DVD general information page
Contains a simple description of DVD, describes the different DVD configurations, and shows an Atomic Force Microscope (AFM) picture of DVD and CD pit structure.
Updated on Jun 29, 1998

Glossaire des termes utilisés par les "Internet Service Providers"

http://isp.webopedia.com/

Glossaire 10



Moving Company
This column on the MonsterMoving tables tells you the name of the company offering a particular program.

Notes
The *Notes* column of the MonsterMoving table may contain any of the following icons...

Indicates that this mover has an ad displayed on the current page.

This mover has an office or local agent located in the specified state.

Jackpot! The listed mover has a partially empty tr uck going your way. Call them immediately to take advantage of this discounted pricing.

Glossaire du déménagement

http://www.moverquotes.com/glossary.html

Thank you for using the CSGNetwork resources! Visit often!

PTA Endorsements

Other awards and honors...

Computer, Telephony & Electronics Industry Glossary

Words beginning with the letters...

A B C D E F G H I J K L M N O

P Q R S T U V W X Y Z OTHER

Or use our local indexed 15,000 word database search to find words anywhere on our entire site...

search

CSGNetwork.Com Site Local Index Only

Select the first letter of the word in question, or for phrases or acronyms beginning with a character other than a letter, select OTHER. You may also use our local indexed search to find words anywhere on our site, not only in the glossary. Clicking on words on the left side of your monitor will take you to those words on that letter page directly. Clicking on Icons in that same column will take you directly to the site represented. They usually depict points of substantial interest or "Icons" of the Internet itself.

Glossaire de l'industrie informatique, téléphonique et électronique

http://www.csgnetwork.com/glossary.html

Glossaire 12

English

- [tech.] **abdomen**
- [pop.] *gut*

Nederlands

- [tech.] **abdomen**
- [pop.] *buik*

Francais

- [tech.] **abdomen (m)**
- [pop.] *ventre*

Glossaire plurilingue médical

http://allserv.rug.ac.be/~rvdstich/eugloss/welcome.html

Glossaire 11

ACCENT

The rhythmically significant stress in the articulation of words, giving some syllables more relative prominance than others. In words of two or more syllables, one syllable is almost invariably stressed more strongly than the other syllables. In words of one syllable, the degree of stress normally depends on their grammatical function; nouns, verbs, and adjectives are usually given more stress than articles or prepositions. The words in a line of poetry are usually arranged so the accents occur at regular intervals, with the meter defined by the placement of the accents within the foot. Accent should not be construed as emphasis.

- **Sidelight: Two degrees of accent are natural to many multisyllabic English words, designated as *primary* and *secondary*.**
- **Sidelight: When a syllable is accented, it tends to be raised in pitch and lengthened. Any or a combination of stress/pitch/length can be a metrical accent.**
- **Sidelight: In English, when the full accent falls on a vowel, as in *PO-tion,* that vowel is called a *long vowel*; when it falls on an articulation or consonant, as in *POR-tion,* the preceding vowel is a *short vowel.* In the classical Greek and Latin quantitive verse, however, *long* and *short* vowels referred to duration, i.e., how long they were held in utterance.**

(See also *Cadence, Ictus, Modulation, Rhythm, Sprung Rhythm, Wrenched Accent*)
(Compare *Caesura*)

CAESURA (siz-YUR-uh)

A rhythmic break or pause in the flow of sound which is commonly introduced in about the middle of a line of verse, but may be varied for different effects. Usually placed between syllables rhythmically connected in order to aid the recital as well as to convey the meaning more clearly, it is a pause dictated by the sense of the content or by natural speech patterns, rather than by metrics. It may coincide with conventional punctuation marks, but not necessarily. A caesura within a line is indicated in scanning by the symbol (||), as in the first line of Emily Dickinson's, *I'm Nobody! Who Are You?*

I'm no | body! || Who are | you?

- **Sidelight: As a grammatical, rhythmic, and dramatic device, as well as an effective means of avoiding monotony, the caesura is a powerful weapon in the skilled poet's arsenal.**
- **Sidelight: Since *caesura* and *pause* are often used interchangeably, it is better to use *metrical pause* for the type of "rest" which compensates for the omission of a syllable.**
- **Sidelight: A caesura occurring at the end of a line is not marked in the scanning process.**
- **Sidelight: The classical caesura was a break caused by the ending of a word within a foot.**

(See *Diaeresis*)
(See also *Alexandrine, Hemistich*)
(Compare *Accent, Cadence, Rhythm*)

ALLITERATIVE VERSE

Poetry in which alliteration is a formal structural element in place of rhyme; it was prevalent in a number of old literatures prior to the 14th century, including Anglo-Saxon. In alliterative verse, the first half-line is united with the second half by alliterating stressed syllables; in the first half-line generally two (but sometimes three) syllables alliterate, while in the second half usually only one. Sometimes one alliterating sound is carried through successive lines:

In a somer seson, whan softe was the sonne,
I shoop me into shroudes as I a sheep were,
In habite as an heremite unholy of werkes,
Wente wide in this world wondres to here.

—*The Vision of Piers Plowman,* by William Langland, 1330?-1400?

- **Sidelight: To facilitate maintaining the alliterative pattern, poets made frequent use of a specialized vocabulary, consisting of many synonymous words seldom encountered outside of alliterative verse.**
- **Sidelight: By the 14th century, rhyme and meter displaced alliteration as a formal element, although alliterative verse continued to be written into the 16th century and alliteration retains an important function as one of a poet's sound devices.**

Glossaire 14

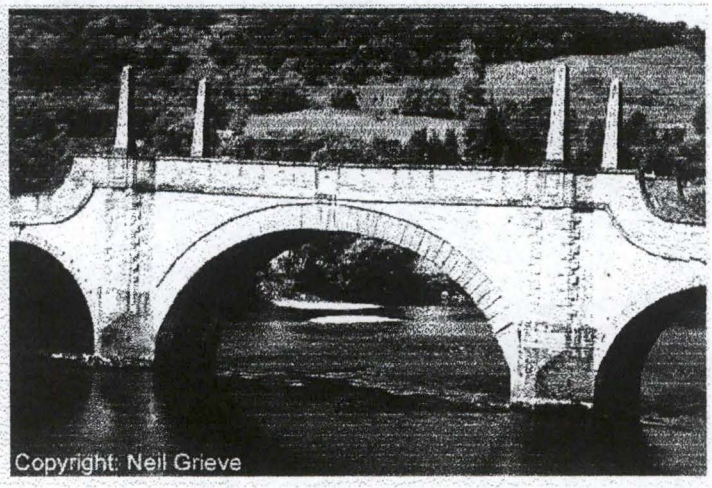
Ancient / Guardianship monument

-(Illustration)

- The Ancient Monuments and Archaeological Areas Act 1979 defines a monument as, "any building, structure or work, whether above or below the surface of the land, and any cave or excavation" or any site comprising the remains of such things or comprising any "vehicle, vessel, aircraft or other movable structure or part thereof." A scheduled monument is any monument included in the schedule to the act. Once scheduled, consent for any works is required from the Secretary of State. The site of a monument includes any adjoining land, which may be considered important to the monument's wellbeing. Most scheduled monuments are **archaeological** sites or ruined **buildings**. Ecclesiastical buildings in use or inhabited buildings cannot be scheduled. Not all ancient monuments are scheduled, the term "ancient monument" actually has a wider meaning which includes both scheduled monuments and any other monument which the Secretary of State decides is of national importance. Buildings can be both scheduled and **listed**, in which case, somewhat controversially, the need to apply for scheduled monument consent takes precedence. Guardianship monuments are monuments in the care of the Secretary of State.

The term "scheduled" derives from the fact that the first ancient monuments were contained in a schedule to the Ancient Monuments Act of 1882. Britain is the only country that operates a separate system for monuments and listed buildings. While controversial, there is a strong presumption that ancient monuments cannot be altered in any way, which avoids confusion which could arise if scheduling and listing were combined. Also, listing cannot be applied to such as caves, ships etc which can enjoy the protection of scheduling.

Wade Bridge at Aberfeldy. Designed by William Adam in 1733 for the ordnance, to the order of Lt. General George Wade. It is both a scheduled monument and a Category A listed building.
Note the centre **arch** ***(detail)*** , framed in **quoined cutwaters** ***(detail)*** and tall obelisks ***(detail)*** .



Glossaire de la conservation des constructions urbaines

http://www.trp.dundee.ac.uk/research/glossary/glossary.html

Glossaire 13

Adenine (A)	Adenin is a purine base (nitrogenous base) and constituent of nulceotides and as such one member of the base pair A-T (adenine-thymine) in DNA and A-U (adenine-uracil) in RNA.

Related Terms:

Purine	A nitrogen-containing, double-ring, basic compound (cf. nitrogenous base) that occurs in nucleic acids. The purines in DNA and RNA are adenine and guanine.
Nitrogenous base	A nitrogen-containing molecule having the chemical properties of a base. See purine and pyrimidine.
Nucleotide	A subunit of DNA or RNA consisting of a nitrogenous base (purine in adenine and guanine, pyrimidine in thymine, or cytosine for DNA and uracil cytosine for RNA), a phosphate molecule, and a sugar molecule (deoxyribose in DNA and ribose in RNA). Depending one the sugar the nucleotides are called deoxyribonucleotides or ribonucleotides. Thousands of nucleotides are linked to form a DNA or RNA molecule. See also base pair.
Base pair (bp)	Two nitrogenous (purine or pyrimidine) bases (adenine and thymine or guanine and cytosine) held together by weak hydrogen bonds. Two strands of DNA are held together in the shape of a double helix by the bonds between base pairs. The number of base pairs is often used as a measure of length of a DNA segment, eg 500 bp.
Thymine (T)	Thymine is a pyrimidine base (nitrogenous base) and constituent of nulceotides and as such one member of the base pair A-T (adenine-thymine) in DNA.
Deoxyribonucleic acid (DNA)	The molecule that encodes genetic information. DNA is a double-stranded molecule held together by weak bonds between base pairs of nucleotides. The four nucleotides in DNA contain the bases: adenine (A), guanine (G), cytosine (C), and thymine (T). In nature, base pairs form only between A and T and between G and C; thus the base sequence of each single strand can be deduced from that of its partner.
Uracil (U)	Uracil is a pyrimidine base (nitrogenous base) and constituent of nucleotides and as such one member of the base pair A-U (adenine-uracil). It is normally found in RNA but not DNA.
Ribonucleic acid (RNA)	A chemical found in the nucleus and cytoplasm of cells; it plays an important role in protein synthesis and other chemical activities of the cell. The structure of RNA is similar to that of DNA. There are several classes of RNA molecules, including messenger RNA, transfer RNA, ribosomal RNA, and other small RNAs, each serving a different purpose.

Glossaire de génétique

http://www.weihenstephan.de/~schlind/genglos.html

Glossaire 16

A - C D - M N - P Q - Z

Acoela("without" "hollow")	an Order of marine turbellarian flatworms characterized by absence of gut and excretory organs. The name refers to the *absence of a gut* and NOT to the absence of a coelom, even though they are also *acoelomate* in that they do lack a *coelom* (see next entry)
Acoelomate ("without" "coelom")	refers to the condition, as seen in flatworms, in which a coelom is absent. The term is applied to triploblasts and not to diploblasts (cnidarians), even though the latter are also "acoelomate". The reason for this is that owing to the lack of mesoderm in diploblasts, a coelom is not possible
Acontium ("javelin or dart")	tubular extensions off the ventral edges of the internal mesenteries in sea anemones which are well-endowed with nematocysts and which function to tangle and disable ingested prey. The acontia generally reside in the gastrovascular cavity but, in some species, they can be protruded out of holes in the body column where they act in defense
Adaptive radiation	refers to the evolutionary spread of organisms into different habitats. Animals most successful at this have been arthropods (especially insects), gastropods, and nematodes
Alternation of generation	a term sometimes used to describe the alternation of sexual (medusa) and asexual (polyp) stages in cnidarians such as hydroids and jellyfish. It is not a useful term for cnidarians because, unlike in mosses and ferns where the expression originated, the two stages in cnidarians often occur simultaneously

Glossaire de zoologie

http://www.zoology.ubc.ca/courses/bio205/

Definitions

Most of the words in this glossary are followed by a phonetic spelling that serves as a guide to pronunciation. The phonetic spellings reflect standard scientific usage and can be easily interpreted following a few basic rules.

- Any unmarked vowel that ends a syllable or that stands alone as a syllable has the long sound. Any unmarked vowel that is followed by a consonant has the short sound.
- If a long vowel appears in the middle of a syllable (followed by a consonant), it is marked with a macron (=). Similarly, if a vowel stands alone or ends a syllable but should have a short sound, it is marked with a breve (u).
- Syllables that are emphasized are indicated by stress marks. A single stress mark (') indicates the primary emphasis; a secondary emphasis is indicated by a double stress mark (").

A

abdomen (ab'duo-men, ab-do'men) The portion of the trunk between the diaphragm and pelvis.

abduction (ab-duk'shun) The movement of a body part away from the axis or midline of the body; movement of a digit away from the axis of the limb.

Glossaire d'anatomie

http://www.mhhe.com/biosci/abio/glossary.mhtml

Densité nutritionnelle (Dn) d'un aliment

C'est la richesse en vitamines et en minéraux d'un aliment par rapport à son apport calorique.
Comparaison entre des céréales complètes et des céréales raffinées

	Fibres en mg /100kcal	Magnésium en mg/100 kcal	Vit B6 en mg/100kcal
Céréales complètes	7	110	0,67
Céréales raffinées	0	28	0 ,15

Les céréales complètes ont donc une densité nutritionnelle plus élevée que les céréales raffinées.

Glossaire du Sport et de la Nutrition

http://www.isostar.tm.fr/nutrition/glossaire/index.html

Glossaire 17

GLOSSAIRE

Protéines

Protéines

Rôle : Elles sont essentielles à la croissance, à l'entretien et au renouvellement cellulaire (musculaire en particulier).

Principales sources :

- Protéines d'origine animale : viandes, oeufs, poissons, lait et fromages.
- Protéines d'origine végétale : soja et dans une moindre mesure, féculents, légumineuses.

Apport énergétique : 1 gramme de protéine fournit 4 kcal ou 17 kJ.
Les protéines animales sont de très bonne qualité nutritionnelle mais elles sont souvent associées à des graisses invisibles et au cholestérol.
Un apport trop important de protéines est inutile car l'organisme détruit l'excédent.
Pour un sédentaire ou un sportif, les protéines doivent représenter 12 à 15 % de l'Apport Energétique Total (AET)

Galerie marchande électronique — **Site web regroupant plusieurs sites de commerce électronique, comme un centre commercial virtuel. L'utilisateur peut commander des biens et services en ligne. *Syn.* Galerie électronique *Angl.* electronic mall, E-mall, *Cf.* commerce électronique.**

GIF — **Abréviation de *Graphic Interchange Format* : Format d'image créé par Compuserve. *Cf.* Image, Compuserve.**

Glossaire des systèmes d'information

http://www.it-seine.com/glossaire/

GLOSSAIRE

PARTIE	Eléments de palettes et caractéristiques
TERME	**charge**
SYNONYME	capacité

Définition

capacité de charge théorique d'une palette, exprimée en kilogrammes, en supposant une charge également et uniformément répartie.

Traductions : 

Glossaire d'une entreprise de construction de palettes

http://www.europal.net/Fr/Infos/I/GlosR/dictionnaire/francais/elements/charge.htm

Glossaire 20

Abcès

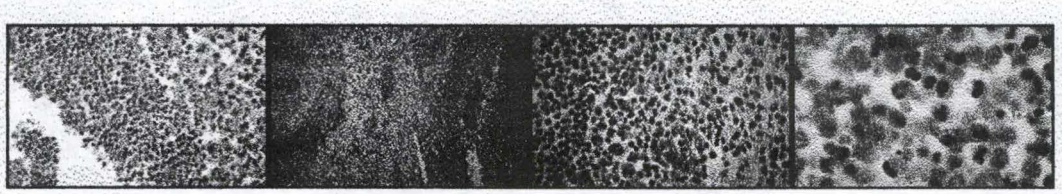
(1537; lat. *abscessus*) (angl. **abscess**)

Lésion inflammatoire constituée par un foyer circonscrit de nécrose tissulaire suppurée (pus) collecté dans une cavité néoformée aux dépens des tissus voisins.

Le terme d'abcès ne doit s'appliquer qu'à une collection purulente développée dans une cavité néoformée au cours de la réction inflammatoire en cause. Lorsque cette cavité est préformée ou naturelle, on devrait utiliser le terme d'empyème. Cette règle est, en pratique, rarement respectée ("abcès sous-phrénique" au lieu de "empyème sous-phrénique").

L'abcès "chaud" (angl. **acute abcess**) est une collection purulente accompagnée de signes cliniques d'inflammation aiguë (chaleur, douleur, tuméfaction). L'abcès "froid" (angl. **cold abscess**) est une collection purulente, de constitution lente, sans signes inflammatoires, pouvant émailler l'évolution de certaines localisations (ostéo-articulaires, ganglionnaires,...) de tuberculose ou de quelques mycoses.

Corrélats : empyème, inflammation purulente, pus, phlegmon.



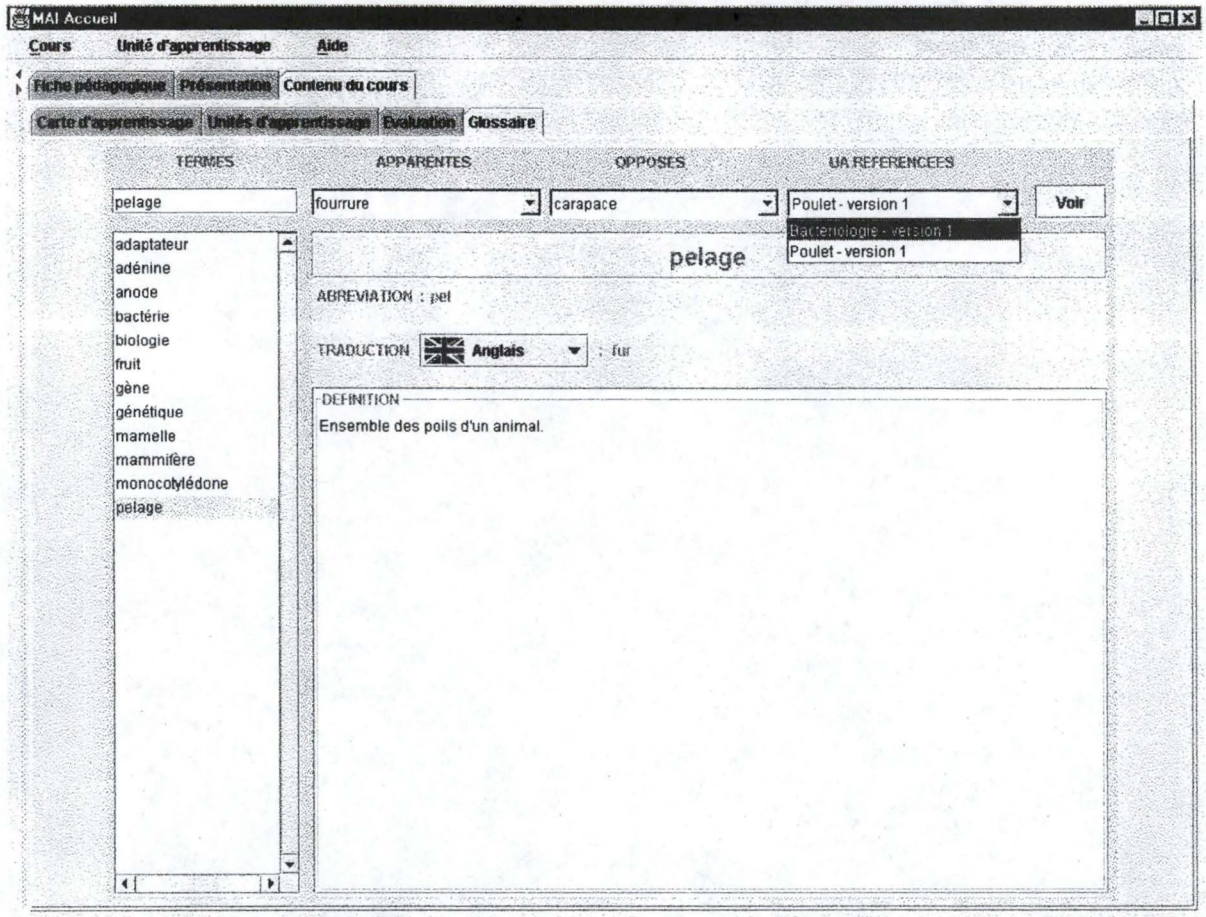
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Glossaire d'anatomie pathologique

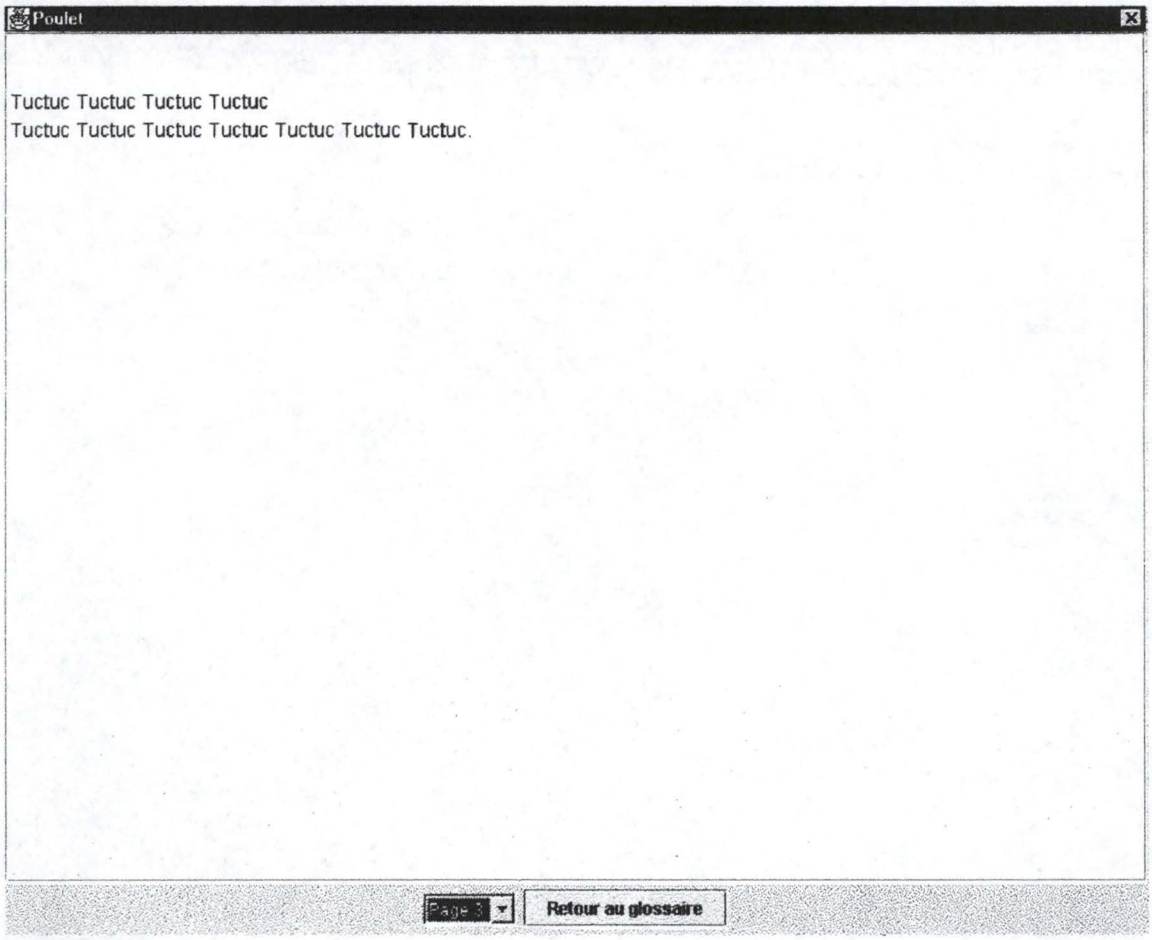
http://www.anapath.necker.fr/enseign/glossaireAP/GlossaireApHomePage

ANNEXE B

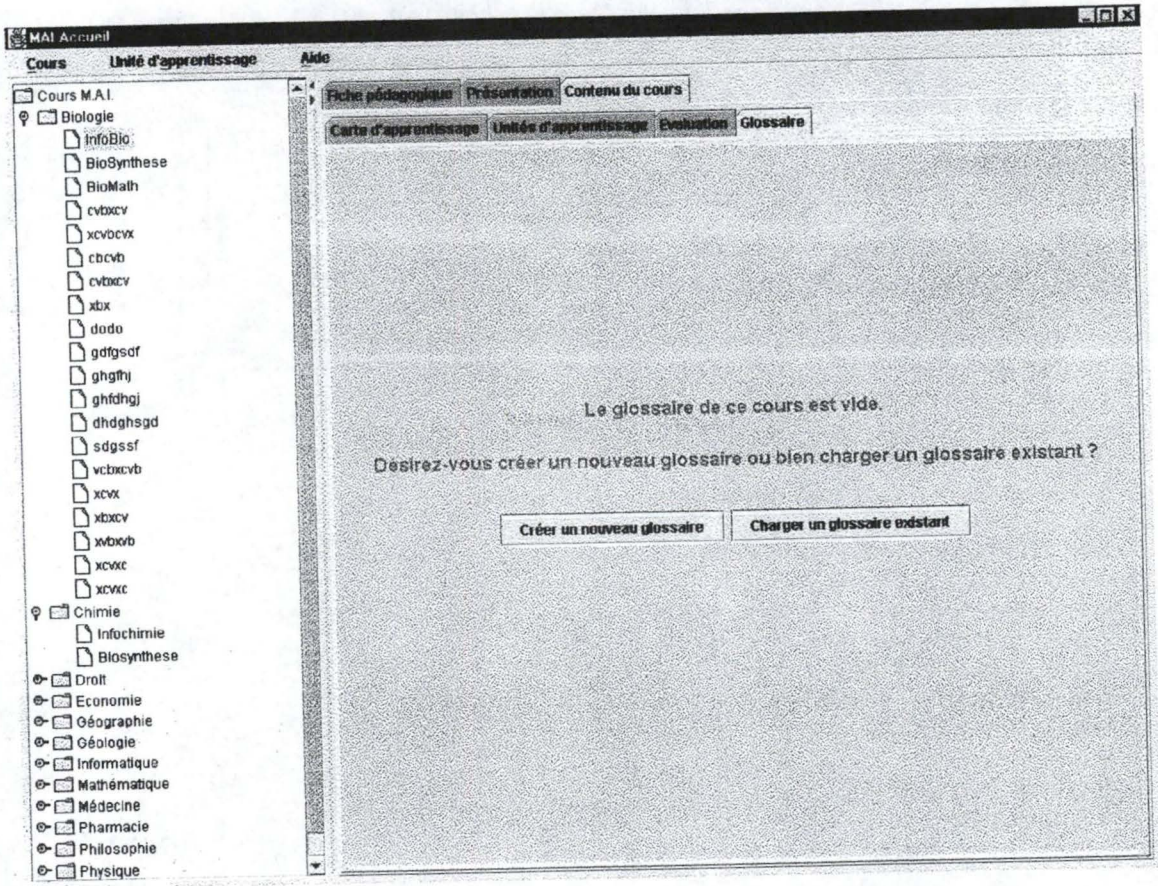
Interfaces du glossaire


MAI Accueil
Cours
Unité d'apprentissage
Aide
Fiche pédagogique
Présentation
Contenu du cours
Carte d'apprentissage
Unités d'apprentissage
Evaluation
Glossaire
TERMES
APPARENTES
OPPOSES
UA REFERENCEES
pelage
fourrure
carapace
Poulet - version 1
Voir
Bactériologie - version 1
Poulet - version 1
adaptateur
adénine
anode
bactérie
biologie
fruit
gène
génétique
mamelle
mammifère
monocotylédone
pelage
pelage
ABREVIATION : pel
TRADUCTION
Anglais
: fur
DEFINITION
Ensemble des poils d'un animal.

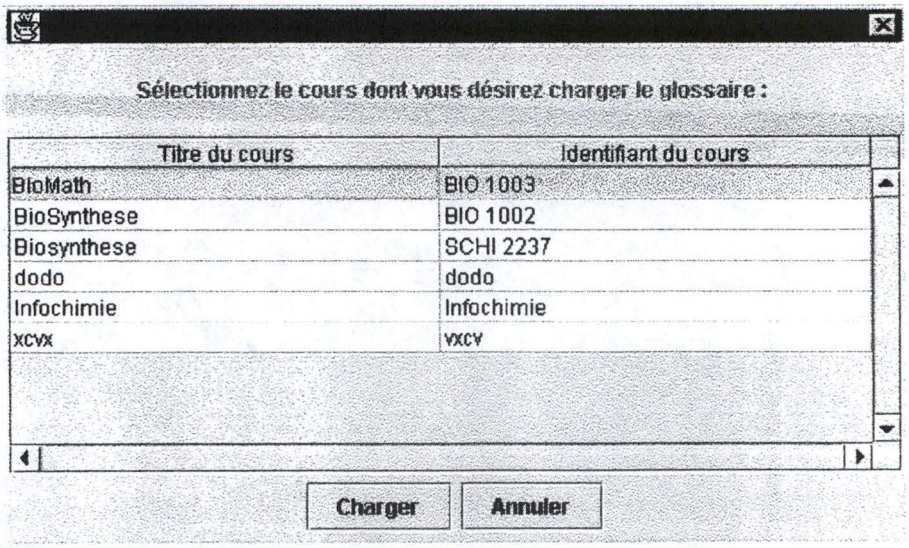
Glossaire


Poulet
Tuctuc Tuctuc Tuctuc Tuctuc
Tuctuc Tuctuc Tuctuc Tuctuc Tuctuc Tuctuc Tuctuc.
Page 3
Retour au glossaire

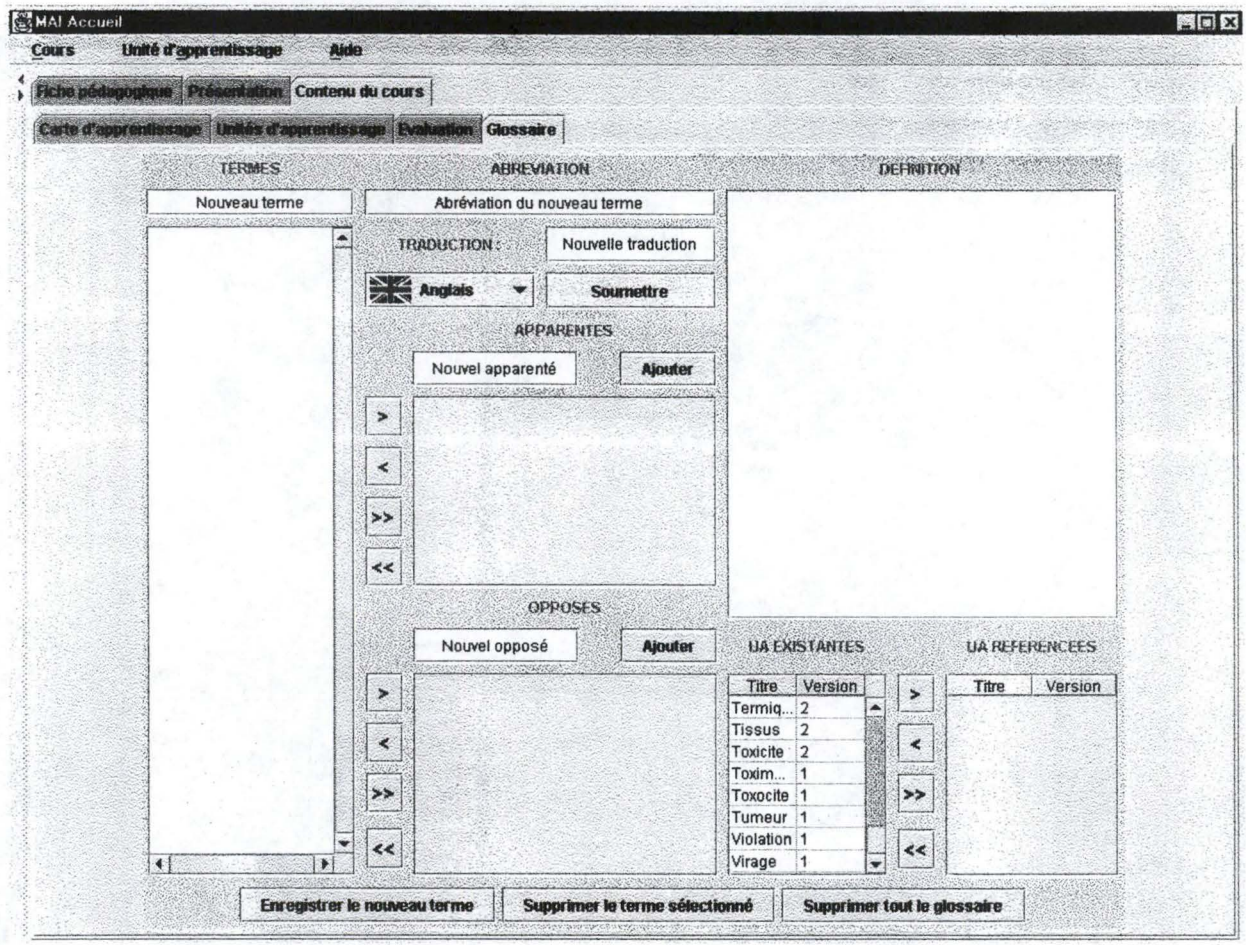
Aperçu UA



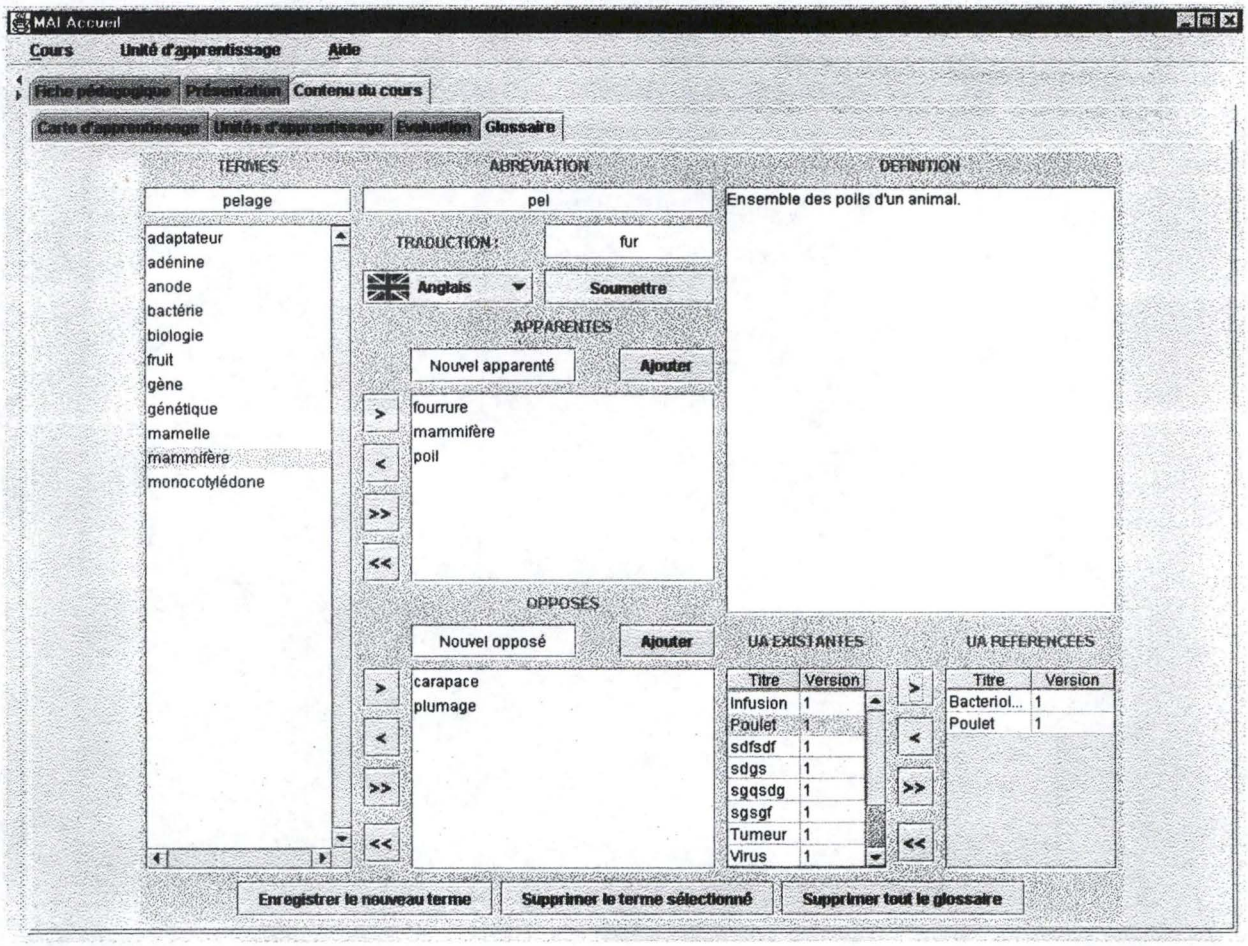
Cas du glossaire vide



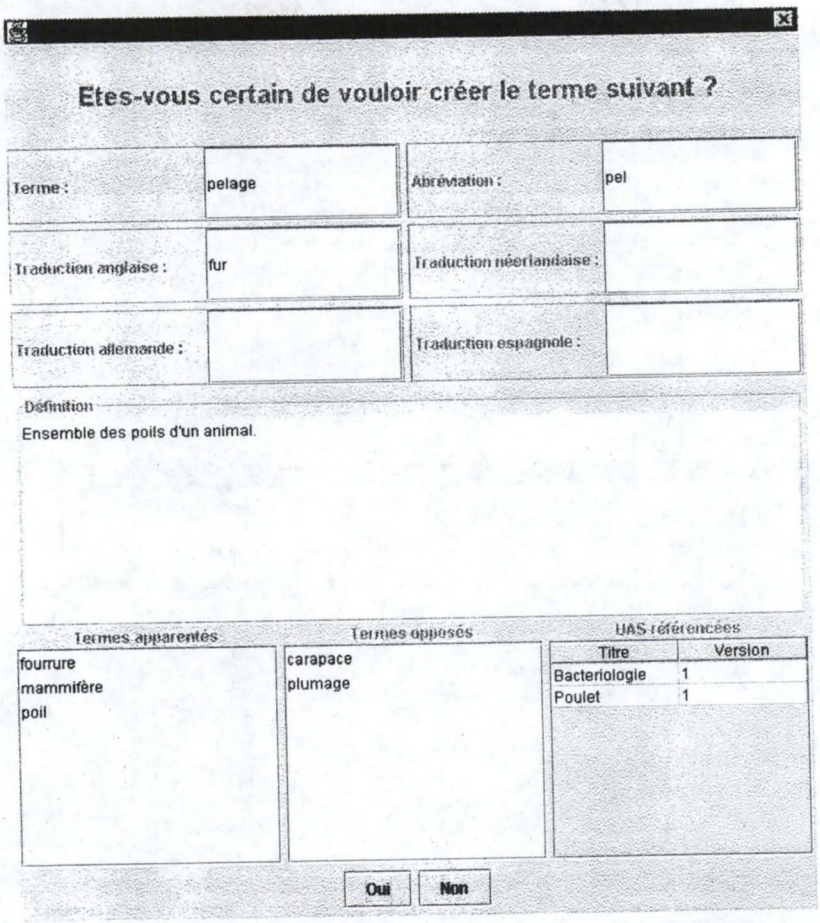
Charger un glossaire



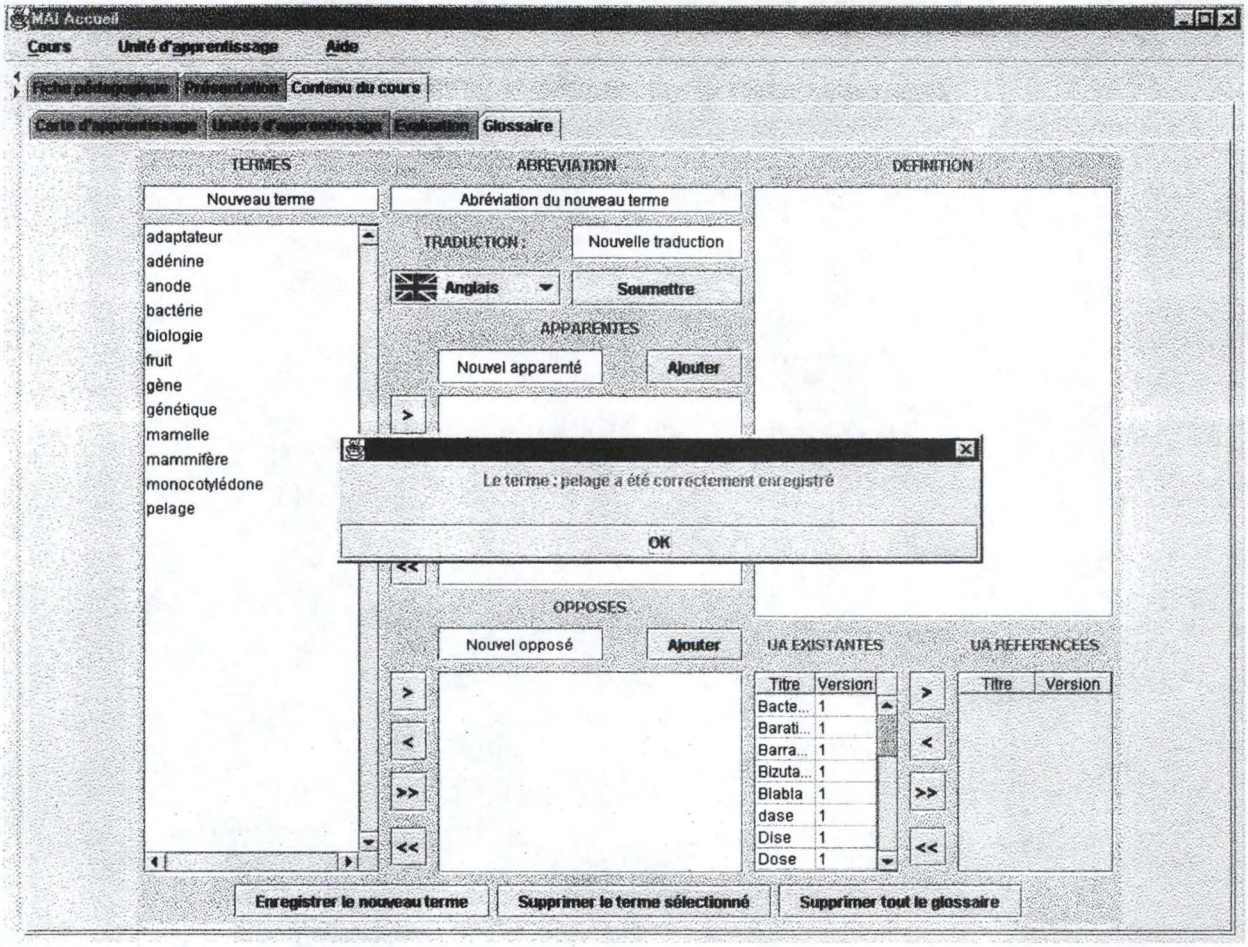
Créer un nouveau glossaire



Création d'un nouveau terme



Récapitulatif de l'enregistrement d'un nouveau terme



Confirmation de l'enregistrement

ANNEXE C

Glossaire[1]

[1] La majorité des définitions se trouvant dans ce glossaire sont tirées de [BODART-PIGNEUR 94], [HAINAUT 94], [LEMAY-CADENHEAD 00], [MILAN 00], [ROBERT 95], [VANDERDONCKT 95] et [VOSS 98].

applet

Programme Java, intégré tel quel dans une page HTML, et qui accomplit une fonction. Ce programme tourne quelle que soit la plate-forme qui le supporte.

application

Le terme "application" est généralement employé pour désigner un logiciel d'application autonome utilisé avec une interface graphique. Une application ne nécessite pas de navigateur Web pour fonctionner. Un programme Java peut être soit une application, soit une applet.

arborescence

("arbre")
Structure logique d'une quantité d'informations organisées de manière hiérarchique et dont la représentation graphique ressemble à un arbre avec ses branches. L'élément le plus élevé dans la hiérarchie s'appelle la "racine" ("root"), les plus bas sont les "feuilles" (fichier), ceux qui sont situés entre les deux sont les "noeuds" (répertoire, noeud).

atomique / composé

Chaque entité d'une table d'une base de données possède des attributs. Ces attributs peuvent être atomiques ou bien composés. Des attributs atomiques sont des attributs insécables.

balise HTML

("tag")
Commande de programmation HTML. Les balises HTML permettent de composer une page Web, de structurer sa présentation. Le principe de base du langage HTML est que tous les éléments (texte, images, sons...) disposés dans une page Web sont placés dans des conteneurs formés par une *<balise ouvrante>* et une *</balise fermante>*.

barre de défilement

("scroll bar")
Contrôle de forme rectangulaire juxtaposé à la droite (gauche) ou au bas (haut) d'une fenêtre ou d'une liste dont la taille ne permet pas de visualiser l'entièreté de l'information. La barre de défilement permet à l'utilisateur de modifier la partie visible de l'information à l'intérieur de la fenêtre ou de la liste.

base de données

Outil permettant de stocker, gérer et consulter des informations, comme des adresses ou des données chiffrées. Ce terme s'applique également aux données primaires enregistrées dans des fichiers. Les données présentes dans une base de données peuvent être triées, filtrées, exploitées et imprimées selon des critères précis.
En plus de la gestion des bases de données proprement dites, les logiciels de base de données sont souvent capables d'utiliser leurs données propres dans des opérations complexes et de fournir les résultats à d'autres systèmes informatiques. Les informations sont généralement stockées dans la base en utilisant une structure fixe, généralement un enregistrement de données (par exemple une adresse), subdivisé en champs tels que : 'Nom', 'Rue', 'Ville'.
Les bases de données servent principalement à interroger des sources d'informations. La réponse apparaît sous la forme d'un compte-rendu ou feuille de réponse. Ces requêtes sont généralement transmises par l'intermédiaire d'un protocole normalisé, tel que SQL ou ODBC, qui permet également d'accéder aux données d'autres logiciels. Les logiciels de bases de données les plus connus sont dBase, FoxPro, MS-Access et Lotus - Approach.

base de données relationnelle

Les bases de données relationnelles, qui sont les plus répandues actuellement, sont constituées de tables. Chaque table est composée d'au moins une colonne dont le domaine est un ensemble de valeurs simples. On peut imposer qu'une colonne ne puisse prendre la valeur *null*. Une table peut avoir un identifiant principal (*primary key*), et plusieurs identifiants additionnels (*unique*). Un ensemble de colonnes peut être utilisé comme référence vers une ligne d'une table (*foreign key*).

canevas

Page à cadres composée de diverses pages HTML organisées dans une même page.

cardinalité Couple de valeurs *min-max* caractérisant un attribut d'un type d'entités ou un type d'association dans lequel ce type d'entités apparaît. Les cardinalités généralement admises sont : *0-1*, *1-1*, *0-N*.

charge de travail Critère ergonomique de conception d'une interface. Une interface est qualifiée d'efficace en charge de travail si et seulement si le volume de données à manipuler et d'actions à accomplir par unité de tâche est réduit.

clé étrangère ("foreign key")
Colonne de référence vers une ligne d'une autre table et qui contient l'identifiant de cette table. Une clé étrangère est constituée de plusieurs colonnes lorsque l'identifiant de la table référencée se compose de plusieurs colonnes.

client L'ordinateur ou bien le programme qui, à l'intérieur d'un réseau, demande des services à un "serveur".

compilateur Programme de traduction d'un code source prélevé dans un fichier source. Le code source, ou programme source, est un programme formulé dans un langage de programmation évolué (par exemple COBOL, FORTRAN, PASCAL, C++, JAVA).
Le texte objet est formulé dans un langage intermédiaire universel, un langage assembleur, ou dans un langage machine.
La compilation s'effectue en trois étapes principales :
1. l'analyse lexicale a pour but de reconnaître les éléments fondamentaux (caractères, nombres, noms et mots-clés) du texte source et de défricher le texte (par exemple en éliminant les commentaires et les caractères superflus) ;
2. l'analyse syntaxique et sémantique permet de vérifier que l'enchaînement des différentes instructions est correct, que les noms des variables ont été correctement employés, que les types de données ont été correctement définis, etc. ;
3. la génération de code restructure les éléments définis au cours de l'analyse syntaxique et sémantique en une suite d'instructions

en langage machine.
La compilation permet d'obtenir un module objet qui n'est pas encore exécutable, car il contient des adresses relatives variables.
C'est un autre programme, l'éditeur de liens, qui transforme le module objet en programme exécutable, en y insérant des sous-programmes standards, des procédures, des fonctions, etc. prélevés dans une bibliothèque de modules objets.
Enfin, un module de chargement convertit les adresses relatives en adresses absolues pour que le programme compilé puisse être effectivement exécuté.

contrainte référentielle

Contrainte selon laquelle la valeur d'une clé étrangère constituée de la colonne CB d'une table B est destinée à désigner une ligne d'une table A.

feuille de calcul

La feuille de calcul d'un tableur se présente généralement sous la forme d'un tableau composé de lignes et de colonnes.

HTML

("HyperText Markup Language")
Langage informatique utilisé pour créer les pages écran de Web sur Internet.

hypertexte

Ensemble de textes et d'autres documents qui peuvent être consultés à partir d'un système de renvois hiérarchisé. La sélection d'un renvoi ou hyperlien permet d'afficher un autre texte, une table, une image...

interface utilisateur

Intermédiaire entre des actions voulues par l'utilisateur pour réaliser sa tâche et la technologie mise à sa disposition.

interpréteur

Programme permettant de traduire le texte source d'un programme écrit dans un langage de programmation évolué dans le langage machine du processeur. Contrairement à un compilateur, la traduction s'effectue pas à pas, c'est-à-dire instruction par instruction. Lorsqu'une instruction a été traduite, elle est immédiatement exécutée.

Le point fort des interpréteurs réside dans leur simplicité. En revanche, comme chaque instruction exécutée plusieurs fois (par exemple dans une boucle) est également traduite plusieurs fois, les programmes interprétés sont relativement lents. Un interpréteur ne peut pas non plus contrôler la compatibilité des séquences d'instructions. Les compilateurs compensent ces points faibles des interpréteurs.

Java Nouveau langage de développement, produit par la société Sun. Ecrit par James Gosling, il permet de doter les documents HTML de nouvelles fonctionnalités : animations interactives, applications intégrées, modèles 3D, etc.
Ce langage est orienté objet et comprend des éléments spécialement conçus pour la création d'applications multimédia.
Pour le moment, Java est disponible pour UNIX, Windows NT et Mac OS et tous les grands fabricants de logiciels ont reçu des licences. Les navigateurs HotJava de Sun (pour le Solaris/X-Windows) et Netscape Navigator 2.0 (pour Windows) sont capables d'interpréter les Applets de Java.

JDBC ("Java DataBase Connectivity")
La technologie JDBC est une bibliothèque de classes offrant la possibilité aux programmes Java de travailler avec une grande diversité de formats de bases de données relationnelles.

liste de choix Composant d'une interface utilisateur qui permet de sélectionner un élément isolé dans une liste déroulante.

liste liée Structure de données servant à stocker une liste séquentielle d'objets. Chaque objet possède une référence vers l'objet qui le suit dans la liste.

métadonnée Donnée qui en décrit une autre.

modèle Entité-Association Mode de description du domaine d'application d'une base de données qui permet de décrire les concepts du domaine, sans

s'attacher à la manière dont ils seront représentés en tables et colonnes. Il offre une représentation graphique des concepts qu'il décrit, ce qui en fait un outil de raisonnement et de spécification attrayant. La description du domaine s'exprimera sous la forme d'un schéma conceptuel.

module

Les ordinateurs personnels sont actuellement construits selon une architecture modulaire. Un "module" est, dans le domaine du matériel informatique, un élément responsable de certaines fonctions internes à l'ordinateur, en association avec d'autres éléments ou avec de grosses unités (par exemple, une carte mère). En raison de sa taille, il peut être remplacé à tout moment par d'autres modules (y compris provenant d'autres fabricants) pour remédier à un défaut ou pour utiliser un module plus performant. Le disque dur, la carte graphique, le processeur, le contrôleur sont des exemples de modules. On utilise aussi les modules dans le domaine des logiciels. Il s'agit de programmes ou de parties de programmes autonomes qui ne remplissent que des tâches bien définies. Certaines procédures peuvent également être des "modules".

monovalué / multivalué

Chaque entité d'une table d'une base de données possède des attributs. Ces attributs sont monovalués si leur cardinalité est *0-1* ou *1-1*. Ils sont multivalués si leur cardinalité correspond à *0-N*.

navigateur

("navigateur de réseau", "browser", "web browser")
Programme qui permet l'accès à World Wide Web sur l'Internet. Les documents du réseau sont visualisés différemment par les divers navigateurs, selon leurs capacités spécifiques. Beaucoup de navigateurs de réseau offrent également la possibilité d'écrire des messages électroniques et d'accéder à des groupes de messages (des recueils de messages, dont toutes les personnes intéressées par certains sujets peuvent s'entretenir). Les navigateurs de réseau les plus connus sont Internet Explorer et Netscape.

objet Unité structurée et limitée. Il est toujours défini par la tâche ou la fonction qu'il accomplit. Il doit contenir tous les éléments dont il a besoin. En informatique, on peut créer un objet, par exemple à l'aide d'un enregistrement composé de champs de données définissant ses propriétés.

ODBC ("Open Data Base Connectivity", "connexion ouverte aux bases de données")
Interface développée par Microsoft en 1992, qui permet l'accès à différents systèmes de bases de données.
Lorsque le pilote correspondant est installé dans le système d'exploitation, l'utilisateur peut accéder à partir d'un programme d'application à diverses bases de données, sans aucune différence.

onglet Echancrure en forme de demi-cercle ou de rectangle dans les feuillets d'un livre pour signaler un chapitre ou une section. En informatique, c'est un élément de l'interface utilisateur qui permet à l'utilisateur de se déplacer d'une partie à l'autre de l'interface.

panneau Conteneur minimal d'une interface utilisateur qui sert à grouper des composants.

pilote Programme ou élément matériel qui va permettre à une machine de fonctionner avec un périphérique particulier.

plate-forme ("plate-forme système" ou "environnement")
Base d'un système logiciel ou matériel, ce dont il a absolument besoin pour exécuter les programmes et pour incorporer les composants matériels.

programmation orientée-objet Forme de programmation qui se veut plus "naturelle". À l'origine de cette nouvelle approche il y a une considération économique : les volumes de codes augmentent rapidement (coûts de production et de maintenance) bien que les problèmes traités soient, dans une large mesure, les mêmes. L'idée était donc de

définir au plus juste les éléments : "objets" et de les inclure dans des familles hiérarchisées : "classes".

La programmation objets présente quatre caractéristiques fondamentales :

1. L'encapsulation : les données et les opérations qui s'y rapportent sont considérées comme une entité ou un objet. D'un point de vue technique, il s'agit d'un "module", avec un "dehors" : liste des services que l'on peut lui demander et un "dedans", privé et inaccessible, sauf conditions spéciales.
2. Le message : il s'agit du nouveau nom de l'appel classique d'une procédure sauf que dans le cas du message, la nature de l'objet détermine les détails de la réponse (qui ne sont pas forcément connus par l'appelant, avec toutes les difficultés de contrôle que cela implique).
3. L'héritage : les objets sont organisés hiérarchiquement dans une arborescence et héritent des données et des opérations de leurs "ancêtres".
4. Le polymorphisme : tous les objets de la hiérarchie peuvent exécuter les "mêmes" fonctions, mais d'une manière adaptée.

prototype

Appareil ou véhicule construit à très peu d'exemplaires, à titre expérimental, ou pour des compétitions.

référence

En programmation, forme d'accès à une valeur. Avec une référence, ce n'est pas à la valeur proprement dite que l'on accède, mais à l'adresse de la valeur.

RMI

("Remote Method Invocation")

Ensemble de protocoles développés par la division JavaSoft de Sun et qui permet aux objets Java de communiquer à distance avec d'autres objets Java. RMI est un protocole relativement simple qui ne travaille qu'avec des objets Java, contrairement aux protocoles plus complexes CORBA et DCOM. Ces derniers sont conçus pour supporter des objets créés en n'importe quel langage.

schéma conceptuel

Modèle mental abstrait qui décrit un modèle Entité-Association et qui n'est pas directement implantable dans un ordinateur, car indépendant des technologies de mise en oeuvre. Il faut donc le traduire en un schéma de base de données sous la forme de tables, de colonnes et de contraintes d'intégrité.

schéma logique

Implémentation d'un schéma conceptuel qui décrit les structures d'une base de données selon le modèle d'une famille de SGBD ("Systèmes de Gestion de Bases de Données") – par exemple : relationnel, orienté-objet...- et telles qu'elles sont perçues par le programmeur ou l'utilisateur du SGBD, mais en évitant les détails techniques propres à un SGBD particulier.

schéma logique relationnel

Schéma logique qui ne contient que des structures et des contraintes directement et explicitement exprimables dans le modèle de SGBD ("Systèmes de Gestion de Bases de Données").

serveur

Il s'agit d'un ordinateur dédié à l'administration d'un réseau informatique. Il gère l'accès aux ressources et aux périphériques et les connexions des différents utilisateurs. Il est équipé d'un logiciel de gestion du réseau : un serveur de fichiers prépare la place mémoire pour des fichiers, un serveur d'impression gère et exécute les sorties sur les imprimantes du réseau et un serveur d'applications rend disponibles sur son disque dur les programmes pouvant être appelés à travers le réseau.

SGBD

("Système de Gestion de Base de Données")
Ensemble des fonctions permettant de formuler des requêtes, de modifier et de stocker des données (traitement de données) ainsi que de définir la structure des données et d'élaborer les fichiers requis (définition de données). Il existe aujourd'hui des systèmes de base de données pour tous les types d'ordinateurs.

socket

Point de connexion à chaque extrémité d'une connexion (TCP, UDP...). Il existe un socket du côté émetteur et un socket du côté récepteur. Un socket TCP est composé de l'adresse IP de

l'ordinateur hôte et du port TCP auquel ce dernier écoute.

SQL ("Structured Query Language", "langage structuré pour les requêtes")
Langage standard développé par IBM pour l'interrogation de bases de données relationnelles.

synchrone/ asynchrone Dans le cadre de la programmation en temps réel, les processus (ou tâches) sont autonomes, mais doivent communiquer entre eux. En mode synchrone, les deux acteurs sont "présents" en même temps (c'est-à-dire que leur déroulement est suspendu pour attendre l'autre) ; en mode asynchrone, les acteurs envoient des messages ou les déposent dans une boîte aux lettres, ou bien consultent un sémaphore (programmation). La première technique est à la fois plus sûre (il n'y a pas d'oubli des nécessaires mises à jour) et plus coûteuse (à cause de l'immobilisation des acteurs).

système auteur Système utilisé pour le développement d'applications multimédia. Les systèmes auteurs fonctionnent en général sur deux niveaux : le niveau auteur et le niveau lecteur. Alors que le niveau auteur comporte toutes les fonctionnalités de création et de présentation, le niveau lecteur peut uniquement lancer et utiliser le programme créé.
En principe, les systèmes auteurs ont une architecture ouverte, c'est-à-dire qu'ils peuvent intégrer différents composants matériels en tant qu'objets et autorisent différents types de traitement (par exemple, le traitement d'images) et de structures de commandes. Alors que les premiers systèmes auteurs devaient être programmés, les systèmes orientés objets, qui ont été développés par la suite, sont utilisés de manière intuitive avec une souris.
Les systèmes auteurs fonctionnent à partir d'une base de données dans laquelle sont gérés les objets et les données. Les systèmes auteurs sont intégrés à l'interface utilisateur et supervisent les interactions avec ce dernier.

table Ensemble de données présentées sous la forme de lignes et de colonnes. Chaque ligne représente une entité, tandis qu'une colonne représente une propriété de ces entités.

TCP/IP ("Transmission Control Protocol/Internet Protocol")
Désigne toute la famille de protocoles qui fut d'abord créée pour le ministère de la défense américaine (Department of Defense - DoD). Leur but était de relier des ordinateurs situés dans différents réseaux (ARPAnet). Aujourd'hui, TCP/IP est utilisé dans de nombreux réseaux locaux et il sert de base au réseau mondial Internet.
Le protocole Internet (IP) s'occupe du transfert des données, tandis que le protocole de contrôle de transmission (TCP) s'occupe de la réception.

téléchargement Action d'envoyer ou de recevoir (upload/download) tout type de données informatiques à partir d'un ordinateur connecté à un réseau.

type d'association Classe de toutes les associations possibles du réel perçu qui vérifient la définition constitutive du type.

UDP ("User Datagram Protocol")
Nom d'un protocole de transmission. IL peut remplacer le TCP. Il n'est pas orienté connexion et n'attend aucune confirmation des différents paquets.

UML ("Unified Modelling Language")
Langage de modélisation dont le but est d'unifier les procédures de conception d'applications orientées objets.

URL ("Uniform Resource Locator")
Adresse standard de n'importe quel document multimédia; sur n'importe quel ordinateur local ou ailleurs dans le WWW.
L'URL a pour structure de base :
protocol:// server/directory/document.

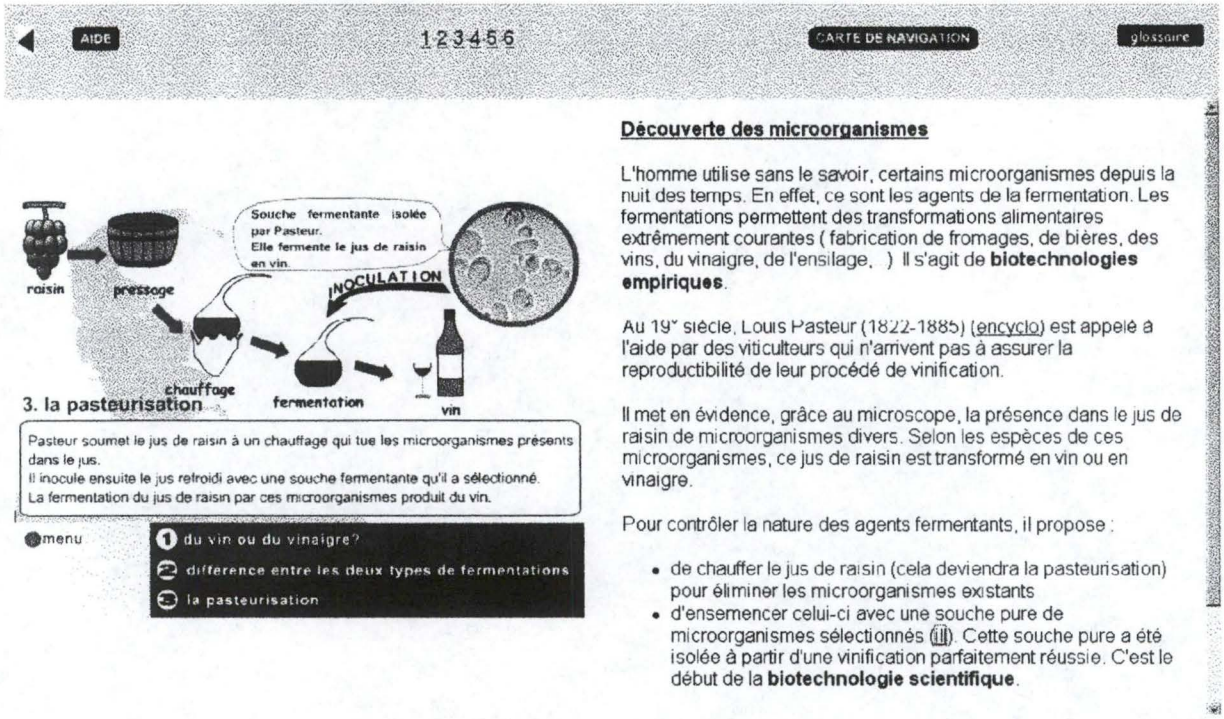
WWW ("World Wide Web")

Système d'information graphique basé sur des liens hypertextes permettant de naviguer d'un site à un autre sur l'Internet.

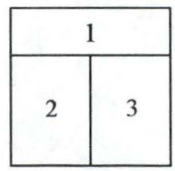
ANNEXE D

Exemple d'une UA réalisée par l'équipe de biologistes

UA intitulée : "Bactéries et levures"



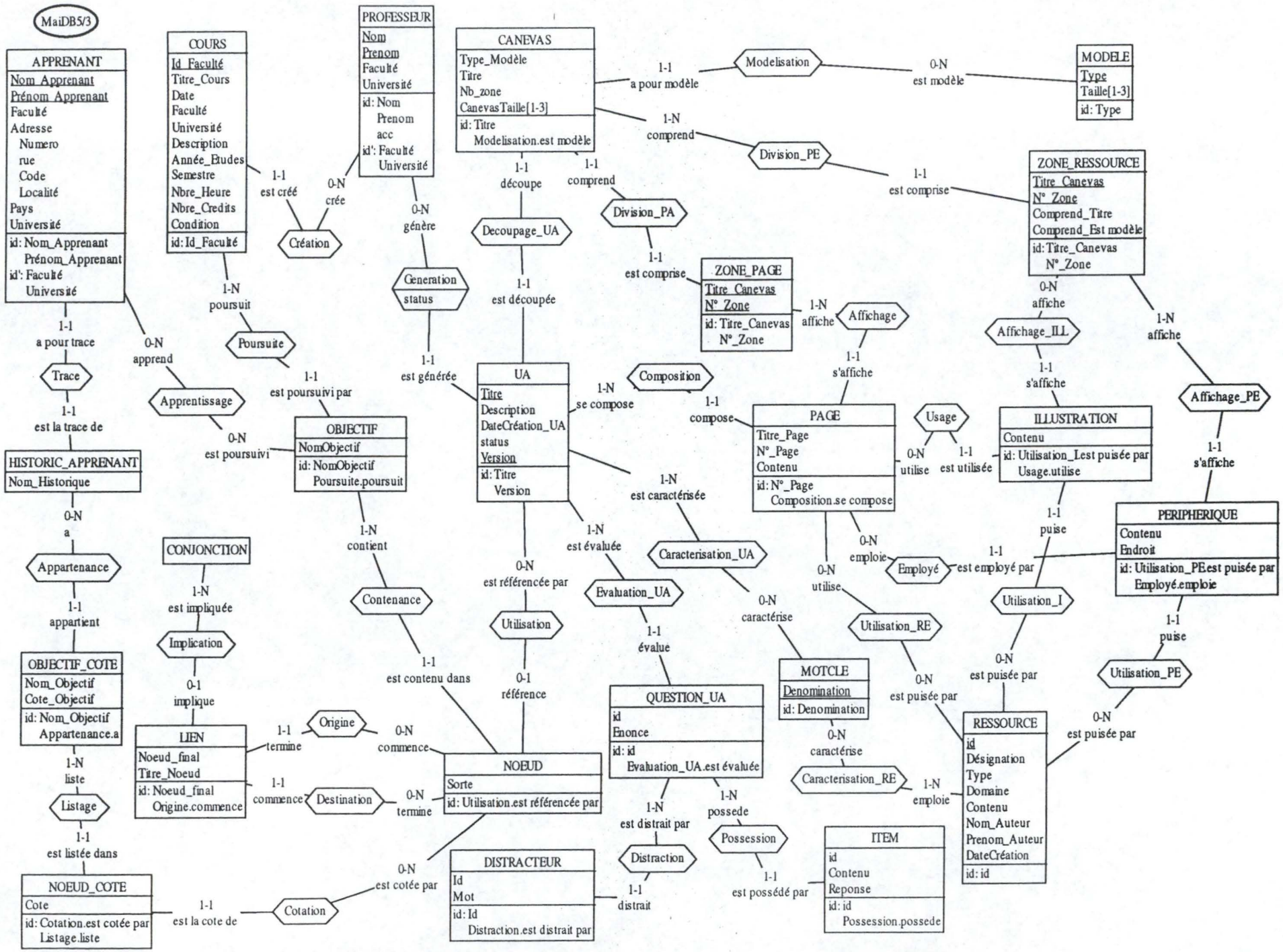
Comme nous pouvons le constater, le canevas choisi pour cette UA est un canevas à 3 zones dont la première zone, qui se trouve tout en haut, contient la barre de navigation ainsi que d'autres fonctionnalités. Les deux autres zones se partagent le reste de l'espace en formant une fenêtre divisée horizontalement en deux parties égales :



La partie de droite est dédiée au texte de l'UA et, la partie de gauche, aux illustrations, animations, etc. Dès que l'utilisateur, au moyen d'un clic de souris sur un lien hypertexte, demande à visualiser l'une de ces ressources périphériques, cette dernière s'affiche dans la partie gauche de l'écran. Dans cet exemple, l'UA choisie dans la carte de navigation (illustrée par la figure 3.2) est "Bactéries et levures". La page de cet exemple est la numéro 2, intitulée "Découverte des micro-organismes". Une barre de navigation de 1 à 6 permet à l'utilisateur de consulter les 6 pages de l'UA de manière linéaire. A ce stade, il est permis à l'utilisateur de faire appel à l'aide, de consulter la carte de navigation ou le glossaire. Dans cet exemple, la zone supérieure de l'écran proposant ces fonctionnalités a été réalisée "artisanalement" par l'équipe de biologistes et sera remplacée par l'outil développé par l'équipe d'informaticiens lorsque les deux volets du système MAI seront intégrés.

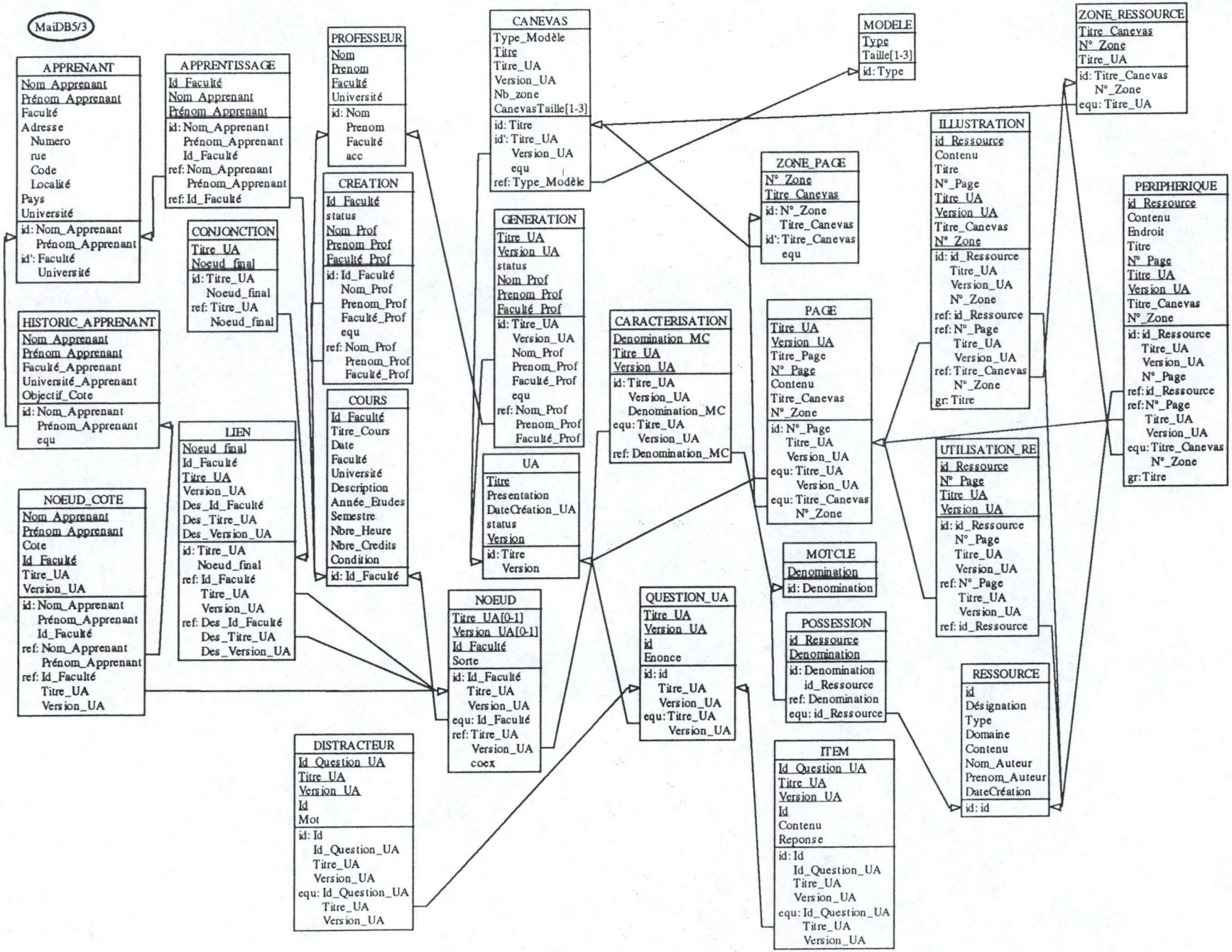
ANNEXE E

Schéma conceptuel de la BD du système MAI


MaiDB5/3
APPRENANT
Nom_Apprenant
Prénom_Apprenant
Faculté
Adresse
Numero
rue
Code
Localité
Pays
Université
id: Nom_Apprenant
Prénom_Apprenant
id': Faculté
Université
COURS
Id_Faculté
Titre_Cours
Date
Faculté
Université
Description
Année_Etudes
Semestre
Nbre_Heure
Nbre_Credits
Condition
id: Id_Faculté
PROFESSEUR
Nom
Prenom
Faculté
Université
id: Nom
Prenom
acc
id': Faculté
Université
CANEVAS
Type_Modèle
Titre
Nb_zone
CanevasTaille[1-3]
id: Titre
Modelisation.est modèle
MODELE
Type
Taille[1-3]
id: Type
ZONE_RESSOURCE
Titre_Canevas
N° Zone
Comprend_Titre
Comprend_Est modèle
id: Titre_Canevas
N°_Zone
ZONE_PAGE
Titre_Canevas
N° Zone
id: Titre_Canevas
N°_Zone
UA
Titre
Description
DateCréation_UA
status
Version
id: Titre
Version
PAGE
Titre_Page
N°_Page
Contenu
id: N°_Page
Composition.se compose
ILLUSTRATION
Contenu
id: Utilisation_I.est puisée par
Usage.utilise
PERIPHERIQUE
Contenu
Endroit
id: Utilisation_PE.est puisée par
Employé.emploie
OBJECTIF
NomObjectif
id: NomObjectif
Poursuite.poursuit
HISTORIC_APPRENANT
Nom_Historique
OBJECTIF_COTE
Nom_Objectif
Cote_Objectif
id: Nom_Objectif
Appartenance.a
CONJONCTION
LIEN
Noeud_final
Titre_Noeud
id: Noeud_final
Origine.commence
NOEUD
Sorte
id: Utilisation.est référencée par
QUESTION_UA
id
Enonce
id: id
Evaluation_UA.est évaluée
MOTCLE
Denomination
id: Denomination
RESSOURCE
id
Désignation
Type
Domaine
Contenu
Nom_Auteur
Prenom_Auteur
DateCréation
id: id
NOEUD_COTE
Cote
id: Cotation.est cotée par
Listage.liste
DISTRACTEUR
Id
Mot
id: Id
Distraction.est distrait par
ITEM
id
Contenu
Reponse
id: id
Possession.possede
1-1
a pour modèle
Modelisation
0-N
est modèle
1-N
comprend
Division_PE
1-1
est comprise
1-1
découpe
Decoupage_UA
1-1
comprend
Division_PA
1-1
est comprise
1-1
est créé
0-N
crée
Création
0-N
génère
Generation
status
1-N
poursuit
Poursuite
1-1
est poursuivi par
1-1
a pour trace
Trace
1-1
est la trace de
0-N
apprend
Apprentissage
0-N
est poursuivi
1-1
est découpée
1-1
est générée
1-N
se compose
Composition
1-1
compose
1-N
affiche
Affichage
1-1
s'affiche
0-N
affiche
Affichage_ILL
1-1
s'affiche
1-N
affiche
Affichage_PE
1-1
s'affiche
Usage
0-N
utilise
1-1
est utilisée
1-N
est caractérisée
Caracterisation_UA
0-N
caractérise
0-N
emploie
Employé
1-1
est employé par
1-1
puise
Utilisation_I
0-N
est puisée par
1-1
puise
Utilisation_PE
0-N
est puisée par
0-N
utilise
Utilisation_RE
0-N
est puisée par
0-N
est référencée par
Utilisation
0-1
référence
1-N
est évaluée
Evaluation_UA
1-1
évalue
1-N
est distrait par
Distraction
1-1
distrait
1-N
possede
Possession
1-1
est possédé par
0-N
caractérise
Caracterisation_RE
1-N
emploie
1-N
contient
Contenance
1-1
est contenu dans
1-N
est impliquée
Implication
0-1
implique
1-1
termine
Origine
0-N
commence
1-1
commence
Destination
0-N
termine
0-N
a
Appartenance
1-1
appartient
1-N
liste
Listage
1-1
est listée dans
0-N
est cotée par
Cotation
1-1
est la cote de

ANNEXE F

Schéma logique relationnel de la BD du système MAI


MaiDB5/3
APPRENANT
Nom_Apprenant
Prénom_Apprenant
Faculté
Adresse
Numero
rue
Code
Localité
Pays
Université
id: Nom_Apprenant
Prénom_Apprenant
id': Faculté
Université
APPRENTISSAGE
Id_Faculté
Nom_Apprenant
Prénom_Apprenant
id: Nom_Apprenant
Prénom_Apprenant
Id_Faculté
ref: Nom_Apprenant
Prénom_Apprenant
ref: Id_Faculté
PROFESSEUR
Nom
Prenom
Faculté
Université
id: Nom
Prenom
Faculté
acc
CANEVAS
Type_Modèle
Titre
Titre_UA
Version_UA
Nb_zone
CanevasTaille[1-3]
id: Titre
id': Titre_UA
Version_UA
equ
ref: Type_Modèle
MODELE
Type
Taille[1-3]
id: Type
ZONE_RESSOURCE
Titre_Canevas
N° Zone
Titre_UA
id: Titre_Canevas
N°_Zone
equ: Titre_UA
CONJONCTION
Titre_UA
Noeud_final
id: Titre_UA
Noeud_final
ref: Titre_UA
Noeud_final
CREATION
Id_Faculté
status
Nom_Prof
Prenom_Prof
Faculté_Prof
id: Id_Faculté
Nom_Prof
Prenom_Prof
Faculté_Prof
equ
ref: Nom_Prof
Prenom_Prof
Faculté_Prof
GENERATION
Titre_UA
Version_UA
status
Nom_Prof
Prenom_Prof
Faculté_Prof
id: Titre_UA
Version_UA
Nom_Prof
Prenom_Prof
Faculté_Prof
equ
ref: Nom_Prof
Prenom_Prof
Faculté_Prof
ZONE_PAGE
N° Zone
Titre_Canevas
id: N°_Zone
Titre_Canevas
id': Titre_Canevas
equ
ILLUSTRATION
id_Ressource
Contenu
Titre
N°_Page
Titre_UA
Version_UA
Titre_Canevas
N° Zone
id: id_Ressource
Titre_UA
Version_UA
N°_Zone
ref: id_Ressource
ref: N°_Page
Titre_UA
Version_UA
ref: Titre_Canevas
N°_Zone
gr: Titre
PERIPHERIQUE
id_Ressource
Contenu
Endroit
Titre
N° Page
Titre_UA
Version_UA
Titre_Canevas
N°_Zone
id: id_Ressource
Titre_UA
Version_UA
N°_Page
ref: id_Ressource
ref: N°_Page
Titre_UA
Version_UA
equ: Titre_Canevas
N°_Zone
gr: Titre
HISTORIC_APPRENANT
Nom_Apprenant
Prénom_Apprenant
Faculté_Apprenant
Université_Apprenant
Objectif_Cote
id: Nom_Apprenant
Prénom_Apprenant
equ
COURS
Id_Faculté
Titre_Cours
Date
Faculté
Université
Description
Année_Etudes
Semestre
Nbre_Heure
Nbre_Credits
Condition
id: Id_Faculté
CARACTERISATION
Denomination_MC
Titre_UA
Version_UA
id: Titre_UA
Version_UA
Denomination_MC
equ: Titre_UA
Version_UA
ref: Denomination_MC
PAGE
Titre_UA
Version_UA
Titre_Page
N° Page
Contenu
Titre_Canevas
N° Zone
id: N°_Page
Titre_UA
Version_UA
equ: Titre_UA
Version_UA
equ: Titre_Canevas
N°_Zone
UTILISATION_RE
id_Ressource
N° Page
Titre_UA
Version_UA
id: id_Ressource
N°_Page
Titre_UA
Version_UA
ref: N°_Page
Titre_UA
Version_UA
ref: id_Ressource
LIEN
Noeud_final
Id_Faculté
Titre_UA
Version_UA
Des_Id_Faculté
Des_Titre_UA
Des_Version_UA
id: Titre_UA
Noeud_final
ref: Id_Faculté
Titre_UA
Version_UA
ref: Des_Id_Faculté
Des_Titre_UA
Des_Version_UA
UA
Titre
Presentation
DateCréation_UA
status
Version
id: Titre
Version
MOTCLE
Denomination
id: Denomination
NOEUD_COTE
Nom_Apprenant
Prénom_Apprenant
Cote
Id_Faculté
Titre_UA
Version_UA
id: Nom_Apprenant
Prénom_Apprenant
Id_Faculté
ref: Nom_Apprenant
Prénom_Apprenant
ref: Id_Faculté
Titre_UA
Version_UA
NOEUD
Titre_UA[0-1]
Version_UA[0-1]
Id_Faculté
Sorte
id: Id_Faculté
Titre_UA
Version_UA
equ: Id_Faculté
ref: Titre_UA
Version_UA
coex
QUESTION_UA
Titre_UA
Version_UA
id
Enonce
id: id
Titre_UA
Version_UA
equ: Titre_UA
Version_UA
POSSESSION
id_Ressource
Denomination
id: Denomination
id_Ressource
ref: Denomination
equ: id_Ressource
RESSOURCE
id
Désignation
Type
Domaine
Contenu
Nom_Auteur
Prenom_Auteur
DateCréation
id: id
DISTRACTEUR
Id_Question_UA
Titre_UA
Version_UA
Id
Mot
id: Id
Id_Question_UA
Titre_UA
Version_UA
equ: Id_Question_UA
Titre_UA
Version_UA
ITEM
Id_Question_UA
Titre_UA
Version_UA
Id
Contenu
Reponse
id: Id
Id_Question_UA
Titre_UA
Version_UA
equ: Id_Question_UA
Titre_UA
Version_UA

ANNEXE G

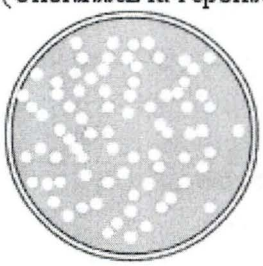
Script SQL de création de la partie de la BD du système MAI consacrée au glossaire

1. Amélioration du questionnaire à choix multiples.

Afin de rendre le cours le plus interactif, les questionnaires à choix multiples présents à la fin de chaque unité d'apprentissage ont été modifiés. Désormais, chaque réponse à une question est précédée d'un bouton radio. L'apprenant peut sélectionner une réponse en cliquant sur le bouton radio correspondant à la réponse choisie.

Un bouton "vérifier !" placé en fin de questionnaire permet de visualiser les bonnes réponses.

Q.6. Sur l'image de la surface d'une boîte de Pétri ci-dessous, chaque zone circulaire est :
(Choisissez la réponse qui vous paraît la meilleure. Une seule réponse est à choisir)



- une seule cellule
- une colonie de quelques cellulee
- un colonies avec beaucoup de cellules
- une colonie sans cellule

Vérifer !

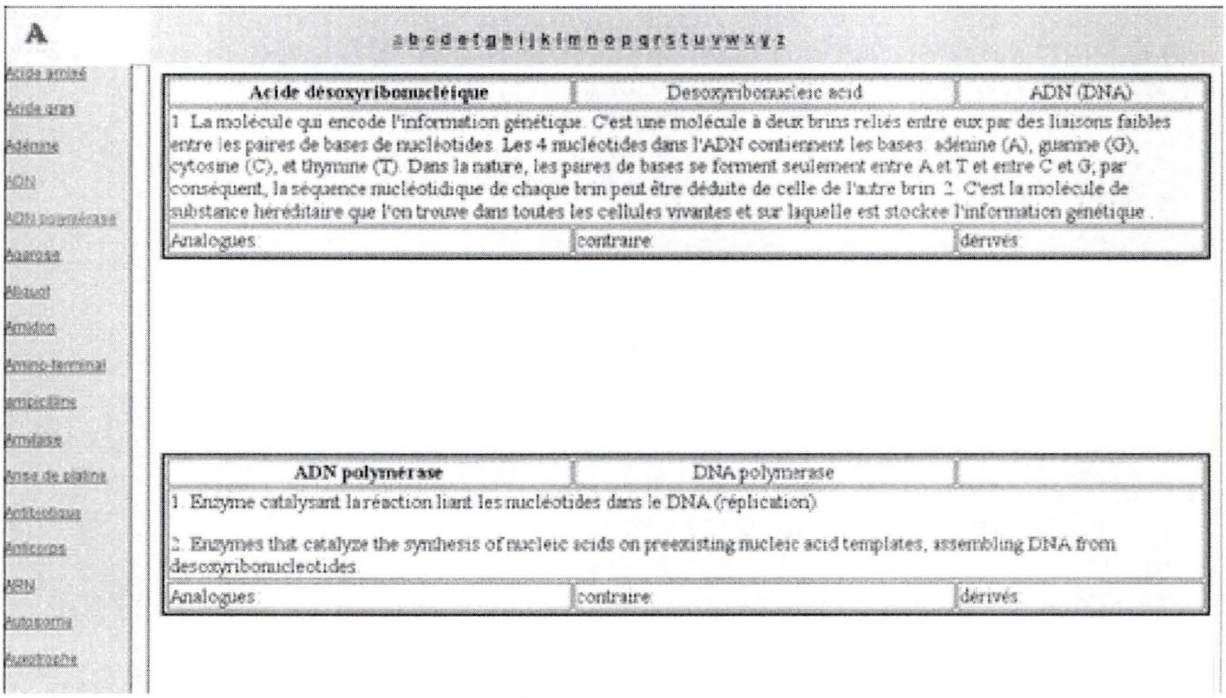
Ces améliorations de présentation et d'interactivité ont été réalisées dans le langage Java Script. Bien que complémentaire du langage HTML, l'éditeur HTML (DreamWeaver 4) ne dispose pas d'outil prêt à l'emploi capable de réaliser un Q.C.M. Le Java Script a été aussi utilisé pour peaufiner la navigation au sein d'une unité d'apprentissage.

Ce langage est une version très allégée du langage JAVA orientée web et correspond parfaitement à nos besoins.
L'utilisation de ce langage nécessite cependant une grande rigueur syntaxique afin d'assurer une portabilité optimale d'une plate-forme à l'autre et d'un navigateur à l'autre (exemple : Netscape est très puriste et n'accepte pas les erreurs de langage alors qu'Internet Explorer est pluslaxiste).
L'étape suivante a été de reformater manuellement les 106 questions (450 réponses) présentes dans le cours de génétique.

La version finale de la partie inclut, en plus des boutons radio, des images et des animations créées pour l'occasion.

2. Glossaire

Tous les liens hypertextes et les définitions présents dans ce cours ont été mis à jour.
La navigation au sein du glossaire a également été retravaillée pour permettre une meilleure lisibilité (chaque définition est mise en évidence par un encadrement bleu sur un fond de couleur jaune pâle) ainsi qu'une plus grande fluidité.



Pas moins de 203 termes ont été implémentés dans le glossaire. Une liste alphabétique les répertorie et un simple clic de souris permet l'affichage d'une définition dans la fenêtre principale.

3. Salon MILIA 2002

Comme lors de sa précédente édition, certains membres de l'équipe MAI (Jacques Van Cleve et Maurice N'Daye Mukuna) ont participé à cet événement pour prendre connaissance des nouveautés dans le domaine du multimédia, établir des nouveaux contacts et aussi pour promouvoir le cours de génétique élaboré dans le cadre de ce projet ainsi que le système auteur MAI. À cette occasion, nous avons :

- Réalisé un poster de présentation illustrant les principales caractéristiques du CD MAI « Programme Multimédia d'apprentissage des principes et des applications du génie génétique ».

- Réalisé des prospectus - carte de visite pour le salon afin promouvoir le CD de génétique. 100 Cartes de visite ont ainsi été réalisées et découpées par nos soins.

- Réalisé un CD de démonstration donnant un aperçu du cours de génétique. 50 CD ont été réalisés et autant de pochettes qui ont du être découpé et inclut dans les boîtiers.

4. Site web

La promotion du cours de génétique nous a conduit à réaliser un site web de démonstration du CD de génétique présentant deux unités d'apprentissage fonctionnelles. Ce site est accessible à l'adresse suivante :

HTTP://WWW.FUNDP.AC.BE/~jvanclev/

Cette réalisation fait suite à de nombreuses demandes de personnes intéressées par notre travail.

5. Répercussion de l'évaluation du cours de génétique

Suite à l'évaluation effectuée au cours du dernier semestre, des améliorations ont été apportées au cours, essentiellement au niveau des images et des animations.

Une attention particulière a été portée à l'allégement des différents répertoires du CD, l'élimination de tous les liens orphelins, des images comme animations désuètes provenant de développement ultérieur du cours, ainsi que les pages obsolètes.

Certains périphériques (animations et illustrations) ont été optimisés pour diminuer le temps nécessaire à leur affichage dans le navigateur, que l'on utilise le CD ou le site WEB. Le travail a consisté à retravailler les périphériques trop volumineux en diminuant leur taille, ou, en décomposant les plus gros documents en une série de plus petits afin de rendre le temps de téléchargement et d'affichage le plus court possible.

Le site, qui occupait précédemment environ 20 Mo, est passé à 12 Mo malgré l'ajout de nouveaux périphériques (images et animations) !

Parallèlement à ces modifications, nous avons procédé à plusieurs corrections de fond détectées lors de l'évaluation. Celle-ci a notamment mis en évidence des problèmes de compréhension dans la dernière unité d'apprentissage du cours. Ces problèmes résultaient pour la plupart d'un manque d'illustrations.

6. Prise de contact avec la maison d'édition Biocosmos.

Plusieurs visites à notre maison d'édition (Ed Biocosmos) à Thuin ont été nécessaire pour présenter le CD de génétique, mettre au point la réalisation et la commercialisation du CD MAI (conception de prospectus, promotion et diffusion du CD de génétique en Belgique et en France).

7. Évaluation du système auteur MAI

Une évaluation du système auteur en développement a été entreprise au cours de ce dernier semestre. Les tests ont porté sur l'insertion de contenus textuels (sans utilisation des ressources périphériques telles que des animations ou des images fixes). Nous avons pu éprouver l'ergonomie et l'aspect pratique de ce programme et mettre en évidence des défauts de conception.

Une collaboration étroite entre la partie informatique et biologique a permis l'amélioration de certaines fonctionnalités ainsi que la nécessité d'en implémenter d'autres. Chaque amélioration a pu être testée, critiquée et améliorée sans délai.

java.sun.com

Advanced Search Search

Technologies
- **J2EE**
- **J2SE**
- **J2ME**
- **XML**
- **Other**

Downloads

Documentation
- **APIs**
- **Tutorials**
- **Code Samples**
- **See All**

Spotlight
- **Web Services**
- **Wireless**

Developer Services
- **Bug Database**
- **Forums**
- **Support**
- **See All**

Java BluePrints

JIMI Software Development Kit

Jimi is a class library for managing images. Its primary function is image I/O. Jimi was formerly a product of Activated Intelligence. Sun is making it available for developers who have code with dependencies on Jimi or for those who need image I/O functionality in applications running under 1.1.x versions of the Java™ Platform. Jimi's range of supported formats includes GIF, JPEG, TIFF, PNG, PICT, Photoshop, BMP, Targa, ICO, CUR, Sunraster, XBM, XPM, and PCX, although some of these formats do not have complete support for all features.

For developers interested in image I/O on the Java 2 Platform, a set of image I/O classes is packaged with the Java Advanced Imaging API (see the com.sun.media.jai.codec package). A standard extension for image I/O on the Java 2 Platform is currently being developed under the Java Community Process.

Download JIMI Software Development Kit:

ZIP Continue

[This page was last updated Jun-20-2002]

Company Info | Licensing | Employment | Press | Help |
JavaOne | Java Community Process | Java Wear and Books

Hierarchy For All Packages

Package Hierarchies:

com.sun.image.codec.jpeg

Class Hierarchy

- class java.lang.**Object**
  - class com.sun.image.codec.jpeg.**JPEGCodec**
  - class com.sun.image.codec.jpeg.**JPEGHuffmanTable**
  - class com.sun.image.codec.jpeg.**JPEGQTable**
  - class java.lang.**Throwable** (implements java.io.Serializable)
    - class java.lang.**Exception**
      - class java.lang.**RuntimeException**
        - class com.sun.image.codec.jpeg.**ImageFormatException**
        - class com.sun.image.codec.jpeg.**TruncatedFileException**

Interface Hierarchy

- interface java.lang.**Cloneable**
  - interface com.sun.image.codec.jpeg.**JPEGDecodeParam**
    - interface com.sun.image.codec.jpeg.**JPEGEncodeParam**(also extends java.lang.Cloneable)
  - interface com.sun.image.codec.jpeg.**JPEGEncodeParam**(also extends com.sun.image.codec.jpeg.JPEGDecodeParam)
- interface com.sun.image.codec.jpeg.**JPEGImageDecoder**
- interface com.sun.image.codec.jpeg.**JPEGImageEncoder**

MAI – Faculté des Sciences – Institut d'informatique – 1999-2003

Nous n'avons pas reçu à temps les informations concernant le projet MAI. Néanmoins, G. Lauters connaît relativement bien le projet puisqu'elle collabore avec l'équipe informatique et l'équipe de biologie depuis plus de 2 ans et a eu l'occasion de tester le prototype. Certains problèmes rencontrés lors du développement de ce projet semblent intéressants et méritaient de figurer dans cette analyse. Dès lors, contrairement aux autres projets présentés ici, ceci ne sera pas une auto-analyse des responsables du projet. Certaines parties de ce descriptif sont tirées d'un mémoire de maîtrise en informatique (en italique)[1].

Description

Le projet MAI (« Modules d'Apprentissages Interactif ») consiste en la réalisation d'un système auteur qui fonctionne à partir du [réseau Internet] *et dont le but premier est d'aider des non-informaticiens à développer des modules de formation. [...] Il permet d'exploiter, d'organiser et de relier différents types de données tels que le texte, l'image et le son de manière à rendre cet ensemble interactif.*

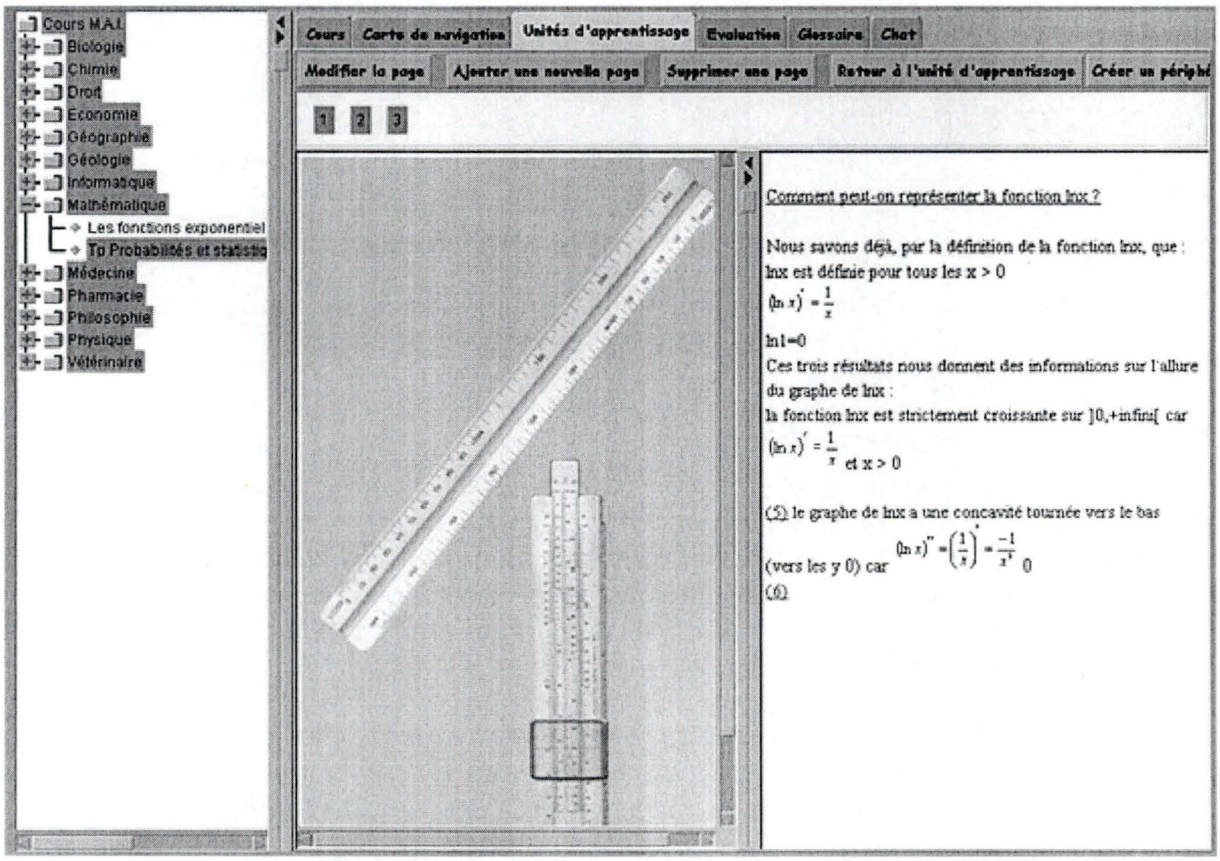


Figure 1 : Capture d'écran du logiciel auteur MAI

Le système auteur fonctionne selon 2 niveaux : celui du professeur et celui de l'apprenant. Le professeur utilise le système auteur pour créer des modules de cours interactifs, des questionnaires d'évaluation, ainsi que la carte de navigation suggérant le parcours sémantique que l'élève doit suivre lors de son apprentissage d'un cours. Le

---

[1] A. Leclerf. 2001. Couplage d'un glossaire avec un système d'apprentissage multimédia. Mémoire de Maîtrise en Informatique. FUNDP.

professeur peut réutiliser des objets existants (modules de cours, illustrations, simulations,...) qu'il va combiner de façon à créer son cours, mais aussi créer personnellement tout nouveaux modules d'apprentissage qu'il assemblera à sa guise. De son côté l'apprenant consulte les modules de cours en respectant le parcours sémantique défini par l'auteur du cours. Il peut également, à la fin de chaque cours, évaluer son niveau d'apprentissage au moyen du questionnaire d'évaluation élaboré par l'auteur du cours. De plus, à tout moment, il peut consulter le glossaire pour obtenir des explications sur un thème précis.

Un autre objectif du projet MAI est de développer un cours *de référence servant de cahier des charges et de test au produit informatique*. Un cours de génie génétique moléculaire a donc été conçu spécifiquement pour le système auteur. Celui-ci a été porté sur CD-ROM (indépendamment du logiciel auteur) et a obtenu le prix Wernaers 2001. Les membres de l'équipe de Biologie participent aussi aux phases d'analyse et de spécification, de validation et de vérification de l'outil informatique.

Le logiciel auteur est encore en stade de développement. A ce jour, le projet MAI a encadré 3 mémoires de fin d'année : 2 en informatique et 1 en mathématique (didactique). Deux cours sont actuellement disponibles : « Mathématique exponentielle » et « probabilités et statistiques ». Certaines parties du cours de génie génétique sont aussi encodées.

Investissement financier et humain

Le budget total du projet MAI est de 743 681 □. Ce financement est assuré par la Région Wallonne.

L'équipe pluridisciplinaire était composée de 2 informaticiens et de 2 scientifiques (Biologie). Pour la dernière année du projet (2002-2003), l'équipe scientifique est réduite à 1 personne.

Options et innovations pédagogiques

L'un des objectifs de ce projet est de « s'éloigner de l'approche linéaire des livres ou syllabi. En effet, le découpage d'un cours en différents modules, organisés selon un réseau sémantique, transforme l'apprentissage linéaire et statique des manuels en un enseignement personnalisé, en fonction des objectifs et acquis de l'apprenant. [...] L'innovation pédagogique apporté par le système MAI, par rapport aux autres produits de formation sur le marché, repose sur cette structure non linéaire, illustrée par la carte de navigation.[...]

Un cours interactif est divisé en une série de problématique ou « objectifs ». L'apprentissage d'un cours se fait par la poursuite des différents objectifs inhérents à ce cours. Chaque objectif regroupe un ensemble de concepts appelés « Unité d'Apprentissage » (UA). [Chaque UA se rapporte à un et un seul concept.] Ces UA sont composées d'un certain nombre de pages [et de ressources périphériques] et sont organisées dans un réseau sémantique, illustré par la carte de navigation.

Choix technologiques

Le système auteur du projet MAI [...] est implémenté entièrement en Java selon une architecture client-serveur. [...] Il a été conçu de manière à alléger au maximum le travail de la [machine utilisateur] en lui assignant les aspects de présentation et le traitement de petites applications. Le serveur de son côté, traite les opération plus complexes et [stocke les

données de chaque cours]. *Les applications [...] sont mises en œuvre grâce à RMI qui permet à des objets Java d'appeler des méthodes d'objets Java s'exécutant sur des ordinateurs distants. [...] Pour communiquer avec la base de données, la technologie JDBC-ODBC* [est utilisée].

Pour le cours de génie génétique auquel s'applique le système auteur de MAI, de nombreuses simulations d'expériences ont été élaborées au moyen du logiciel Macromedia Flash. Les illustrations ont été développées sous Macromédia Freehand et le cours a été agencé à l'aide du logiciel Macromedia Dreamweaver. L'interactivité à été mise en œuvre grâce au langage JavaScript.

Difficultés rencontrées

Informatique

Les principales difficultés rencontrées se situent au niveau du développement du logiciel auteur. Tout d'abord, le choix du langage informatique initial, maîtrisé par la première équipe informatique avant le début du projet, n'était pas des plus judicieux. En effet, ce langage (Visual Basic) n'était absolument pas portable puisqu'il se limitait à l'environnement Windows. Le projet a donc été réorienté après 1 an de développement vers le langage Java, *a priori* multiplateformes. Cependant, la version du langage choisi (Java 2) n'est, à ce jour, toujours pas implémenté dans l'univers Mac. Il aurait peut être été opportun de choisir une version antérieure, qui avait fait ses preuves, et qui fonctionnera encore longtemps. Cependant, la version 2 permettait de développer plus facilement certaines fonctionnalités du logiciel auteur et est beaucoup plus stable que les versions précédentes. En tout état de cause, à l'époque, le langage Java était extrêmement prometteur et il était inconcevable que plusieurs années après, Apple n'ait toujours pas implémenté la dernière version. Dès lors, à l'heure actuelle, le logiciel MAI ne fonctionne que dans l'environnement Windows. Il est cependant possible de le faire tourner sur un Mac en émulant Windows grâce au logiciel Virtual PC. Une autre technique impliquant le « MacOS for Java » est à l'étude mais elle nécessite aussi d'installer des composants sur la machine cliente.

Par ailleurs, l'équipe informatique a changé plusieurs fois en cours de projet, avec tous les retards de développement que cela peut occasionner. Elle est composée de jeunes informaticiens qui ont dû apprendre à gérer les contraintes d'un projet d'une telle envergure et se former au langage et technologies choisies, et ceci sur toute la durée du projet, ce qui a nécessairement retardé l'échéance du projet.

Biologie

En ce qui concerne le cours de biologie, il était prévu, initialement, que le cours soit directement encodé dans le logiciel MAI. Cependant, le découpage du cours, la conception et la réalisation des simulations, animations et illustrations furent en grande partie achevées, de même qu'une première ébauche des fonctionnalités de l'interface apprenant, avant qu'un prototype fonctionnel soit disponible. L'équipe de biologie a donc développé le cours sous Dreamweaver (éditeur de page Web) afin de pouvoir le tester. Ceci a aboutit à la production d'un CD-ROM de biologie moléculaire, complètement indépendant du logiciel MAI (en dehors de la philosophie de découpage en unité d'apprentissage et de navigation à travers une carte de concepts) qui a d'ailleurs obtenu le prix Waerners 2001.

C'est pourquoi, à l'heure actuelle, on peut dire que le projet MAI se compose en fait de 2 produits : le CD-ROM de biologie moléculaire et le logiciel auteur. Le CD-ROM aurait parfaitement pu être développé indépendamment. Son contenu sera cependant intégré au logiciel auteur comme prévu initialement.

Les problèmes rencontrés lors du développement de ce CD-ROM sont classiques : problèmes de compatibilités entre navigateurs et plateformes différentes, problème d'évolution de l'ergonomie (à l'époque les *frames* était à la mode et le cours à été conçu avec ce système, à l'heure actuelle cette technique est déconseillée par les ergonomes)…

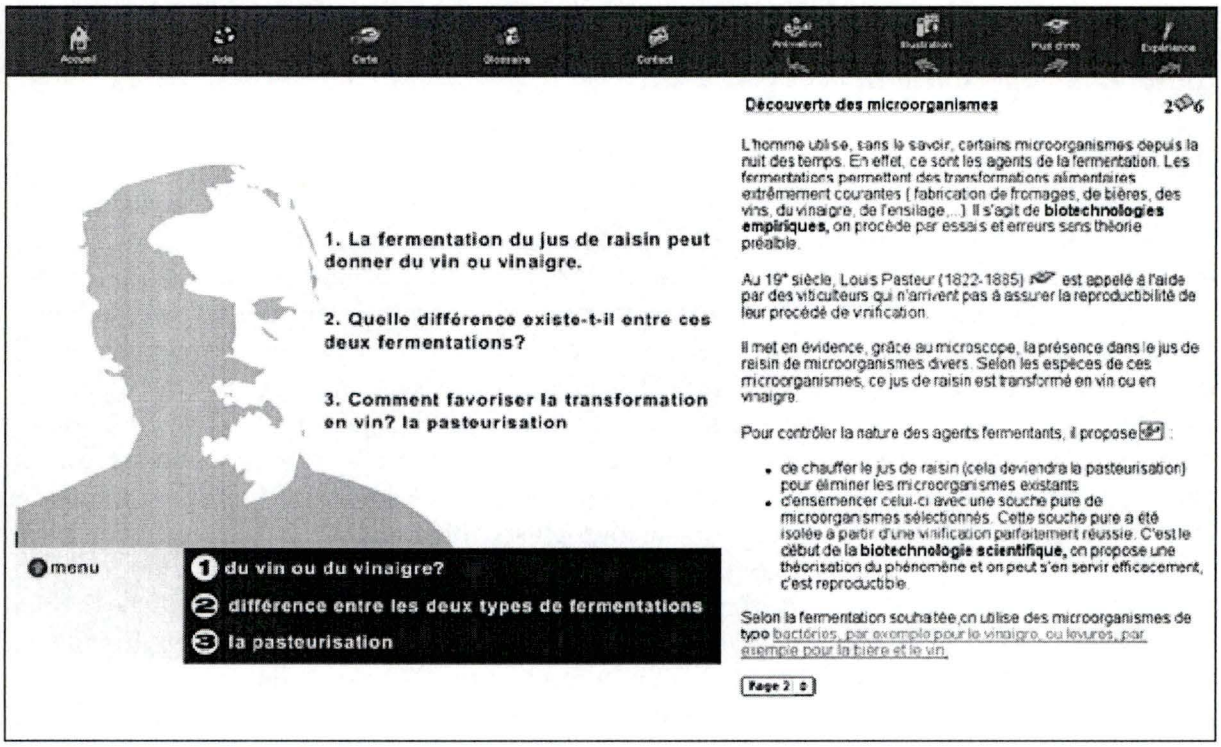


Figure 2 : Capture d'écran du CD-ROM de génie génétique

Un autre problème est à prendre en compte en ce qui concerne l'équipe scientifique. La personne actuellement engagé est arrivé bien après toutes les phases de réflexion, de conception, de découpage du contenu, de développement des animations,… Sa tâche a été de terminer le prototype afin de presser les Cd-ROM (finaliser l'interface et la navigation, optimiser les illustrations et animations, ajouter de l'interactivité, le packaging…), d'assurer la promotion du CD-ROM lors des salons professionnels, d'encoder les cours dans le logiciel auteur et de le tester pour l'équipe informatique. Ce travail (en particulier les 2 dernières tâches) n'est pas vraiment valorisant pour un scientifique.

Général

La communication entre les 2 équipes et leurs responsables n'a pas toujours été aisée. Tout d'abord, initialement, les 2 équipes (biologistes et informaticiens) étaient géographiquement séparées. A leur demande les scientifiques et les informaticiens ont été réunis dans un même local. Ceci a permis une meilleure réflexion informelle en temps réel, les informaticiens pouvant mieux répondre aux besoins des biologistes qu'au travers des réunions régulières. Ensuite le problème classique de communication dû à la différence de « langage » entre personnes de spécialités différentes a obligé chacun à vulgariser pour l'autre son domaine de compétence.

Points positifs

L'un des informaticiens estime que ce projet d'envergure est un challenge très intéressant pour un jeune diplômé. En effet, il lui semble très motivant de développer, de prendre des décisions stratégiques de développement et de mener à son terme un produit qui a pour ambition d'être utile à la communauté universitaire. Ce projet lui a permis de diversifier ses compétences et ses connaissances. Il estime néanmoins qu'il faut prendre en compte le temps de formation des informaticiens pour respecter les échéances.

L'un des biologistes considère, quant à lui, que ce projet lui a beaucoup apporté, que ce soit au niveau intellectuel (réflexion sur la conception d'un cours multimédia, découpage) ou informatique. Il a en effet appris seul à maîtriser des logiciels de dessins, d'animations, de conception Web et a eu carte blanche pour développer ces ressources. Cette activité lui a beaucoup plu. A la suite de ce projet il a continué dans cette voie en développant des rappels théoriques de Biostatistique et des simulations pour les TP. Grâce à tous ces acquis, il travail actuellement sur une thèse en didactique.

Enfin la collaboration entre les 2 équipes (informaticiens et scientifiques) est perçue comme extrêmement profitable à l'ensemble du projet, de même que la collaboration avec le chercheur de l'UAM qui a beaucoup apporté au projet (que ce soit pour la partie informatique ou pour le développement du Cd-ROM de biologie).

Évaluation

Une évaluation du CD-ROM de génie génétique par des membres de la Faculté des Sciences, des enseignants du secondaire et des étudiants en régendat a été faite. Néanmoins, les résultats ne sont pas encore analysés.

Conclusions

```
-- ********************************************
-- * Standard SQL generation                  *
-- *------------------------------------------*
-- * Generator date: Feb 22 1999              *
-- * Generation date: Tue Aug 14 17:57:58 2001 *
-- ********************************************


-- Database Section
-- ________________

create database GlossarySchema;


-- DBSpace Section
-- _______________


-- Table Section
-- _____________

create table ANTONYM (
    NAME varchar(50) not null,
    COURS varchar(15) not null,
    A_NAME varchar(50) not null,
    primary key (NAME, COURS, A_NAME));

create table CHARACTERIZATION (
    NAME varchar(50) not null,
    COURS varchar(15) not null,
    TITRE varchar(1) not null,
    SHORT_NAME char(1) not null,
    VERSION numeric(1) not null,
    primary key (NAME, COURS, VERSION, TITRE));

create table COURS (
    ID_FACULTE varchar(15) not null,
    NOM_PROF varchar(15) not null,
    PRENOM_PROF varchar(15) not null,
    FACULTE_PROF varchar(15) not null,
    TITRE_COURS varchar(15) not null,
    DATE date not null,
    FACULTE varchar(15) not null,
    UNIVERSITE varchar(15) not null,
    DESCRIPTION varchar(255) not null,
    ANNTE_ETUDES numeric(1) not null,
```

```
    SEMESTRE numeric(1) not null,
    NBRE_HEURE numeric(1) not null,
    NBRE_CREDITS numeric(1) not null,
    CONDITION varchar(15) not null,
    primary key (ID_FACULTE));

create table RELATEDTERMS (
    NAME varchar(50) not null,
    COURS varchar(15) not null,
    R_NAME varchar(50) not null,
    primary key (NAME, COURS, R_NAME));

create table TERM (
    NAME varchar(50) not null,
    COURS varchar(15) not null,
    ABBREVIATION char(1) not null,
    ENGLISHTRANSLATION varchar(50) not null,
    DUTCHTRANSLATION char(1) not null,
    GERMANTRANSLATION char(1) not null,
    SPANISHTRANSLATION char(1) not null,
    DEFINITION varchar(255) not null,
    primary key (NAME, COURS));

create table UA (
    TITRE varchar(1) not null,
    PRESENTATION varchar(255) not null,
    DATECRTATION_UA date not null,
    STATUS varchar(1) not null,
    VERSION numeric(1) not null,
    FACULTE char(1) not null,
    COURS char(1) not null,
    CONTENU_DE_PRESENTATION char(1) not null,
    primary key (TITRE, VERSION));


-- Constraints Section
-- _____

alter table ANTONYM add constraint FKisantonymof_1
    foreign key (NAME, COURS)
    references TERM;

alter table CHARACTERIZATION add constraint CHARACTERIZATION
    foreign key (NAME, COURS)
    references TERM;
```

```
alter table CHARACTERIZATION add constraint CHARACTERIZATION_1
    foreign key (TITRE, VERSION)
    references UA;

alter table RELATEDTERMS add constraint FKisrelatedto_1
    foreign key (NAME, COURS)
    references TERM;

alter table TERM add constraint TERM
    foreign key (COURS)
    references COURS;


-- Index Section
-- _____

create unique index IDANTONYM
    on ANTONYM (NAME, COURS, A_NAME);

create index FKisantonymof_1
    on ANTONYM (NAME, COURS);

create unique index IDCHARACTERIZATION
    on CHARACTERIZATION (NAME, COURS, VERSION, TITRE);

create index CHARACTERIZATION
    on CHARACTERIZATION (NAME, COURS);

create index CHARACTERIZATION_1
    on CHARACTERIZATION (TITRE, VERSION);

create unique index IDCOURS
    on COURS (ID_FACULTE);

create unique index IDRELATEDTERMS
    on RELATEDTERMS (NAME, COURS, R_NAME);

create index FKisrelatedto_1
    on RELATEDTERMS (NAME, COURS);

create unique index IDTERM
    on TERM (NAME, COURS);

create index TERM
    on TERM (COURS);
```

```
create unique index IDUA
    on UA (TITRE, VERSION);
```

ANNEXE H

Code Java du glossaire du système MAI

```
1  //Title:        ClientLocal
2  //Version:      1
3  //Copyright:    Copyright (c) 2001
4  //Author:       Audrey Lecerf
5  //Company:      MAI PROJECT - FUNDP
6  //Description:  Intermédiaire où l'on décidera d'afficher Teacher ou Student en fonction
                   de la catégorie de l'utilisateur
7
8  import java.rmi.*;
9  import java.rmi.registry.*;
10 import java.net.*;
11 import javax.swing.*;
12
13 public class ClientLocal
14 {
15 String categorie;
16 String cours;
17 public JPanel panel = new JPanel();
18
19 public ClientLocal(String cat, String cou)
20     {
21     categorie = cat;
22     cours = cou;
23     ServerRemote serveur;
24     Registry repertoire;
25
26     try
27     {
28     /* Accès au serveur distant */
29
30     repertoire = (Registry)LocateRegistry.getRegistry("milan.info.fundp.ac.be");
31     serveur = (ServerRemote)repertoire.lookup("glossary");
32
33     /* Lancer l'interface Professeur ou Apprenant en fonction de la catégorie */
34
35     if (categorie.equals("Apprenant")) panel = new Student(serveur, cours);
36     else if (categorie.equals("Professeur")) panel = new Teacher(serveur, cours);
37     else new Msg("Catégorie invalide");
38     }
39
40     catch (RemoteException re)
41     {
42     re.printStackTrace();
43     }
44
45     catch (NotBoundException nbe)
46     {
47     nbe.printStackTrace();
48     }
49     }
50 }
```

```
1  //Title:        ComboBoxRenderer
2  //Version:      1
3  //Copyright:    Copyright (c) 2001
4  //Author:       Audrey Lecerf
5  //Company:      MAI PROJECT - FUNDP
6  //Description:  Traite l'aspect de la JComboBox : image + texte
7
8  import java.awt.*;
9  import java.awt.event.*;
10 import javax.swing.*;
11
12 class ComboBoxRenderer extends JLabel implements ListCellRenderer
13     {
14         public ComboBoxRenderer()
15         {}
16
17         public Component getListCellRendererComponent(
18                                             JList list,
19                                             Object value,
20                                             int index,
21                                             boolean isSelected,
22                                             boolean cellHasFocus)
23         {
24         ImageIcon icon = (ImageIcon)value;
25         setText(" " + icon.getDescription());
26         setIcon(icon);
27         return this;
28         }
29     }
```

```
1  //Title:        CreateUasVector
2  //Version:      1
3  //Copyright:    Copyright (c) 2001
4  //Author:       Audrey Lecerf
5  //Company:      MAI PROJECT - FUNDP
6  //Description:  Classe qui transforme un vecteur de noms d'UAS en un vecteur
                   contenant les UAS réelles
7
8
9  import java.awt.*;
10 import javax.swing.*;
11 import java.util.*;
12 import java.io.*;
13 import java.lang.Object.*;
14 import java.awt.event.*;
15 import java.sql.*;
16 import java.sql.Connection.*;
17 import java.rmi.*;
18
19 import javax.swing.event.*;
20 import javax.swing.border.*;
21
22
23 class CreateUasVector implements Serializable
24 {
25 ServerRemote serv;
26 Connection con;
27 Vector nom_uas = new Vector();
28 Vector uas = new Vector();
29
30 public CreateUasVector(Vector nom_uas, ServerRemote serv) throws RemoteException
31     {
32     this.nom_uas = nom_uas;
33     this.serv = serv;
34
35     try
36         {
37         Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
38         con = DriverManager.getConnection("jdbc:odbc:bdmai", "sa", "xxxxxx");
39         DatabaseMetaData dma = con.getMetaData();
40         }
41
42    catch (Exception e)
43         {
44         e.printStackTrace();
45         }
46
47     loadUAS(nom_uas);
48
49     try
50         {
51         con.close();
52         }
53
54     catch(Exception e)
55         {
56         e.printStackTrace();
57         }
58     }
59
60  void loadUAS(Vector noms) throws RemoteException
61     {
62     int size = noms.size();
63     int x = 0;
64
65     while (x<size)
66         {
67         String t = (String)((Vector)(noms.get(x))).get(0);
68         int v = (int)Integer.parseInt((String)((Vector)(noms.get(x))).get(1));
69         UALoading ual = new UALoading(t, v, serv, con);
70         uas.add(ual.ua);
71         x++;
72         }
73     }
74
75 }
```

```
1  //Title:        DB
2  //Version:      1
3  //Copyright:    Copyright (c) 2001
4  //Author:       Audrey Lecerf
5  //Company:      MAI PROJECT - FUNDP
6  //Description:  Exécution de la query sur la bd et transformation du ResultSet en Vector
7
8  import java.sql.*;
9  import java.sql.Connection.*;
10 import java.util.*;
11 import java.io.*;
12
13 public class DB implements Serializable
14 {
15 public Vector getData(String query, Connection con)
16     {
17         Vector data = new Vector();
18         Statement stat = null;
19         ResultSet result = null;
20
21         try
22             {
23             stat = con.createStatement();
24             result = stat.executeQuery(query);
25             ResultSetMetaData rsmd = result.getMetaData();
26             int numCol = rsmd.getColumnCount();
27
28             /* Transformation du ResultSet en Vector */
29
30             while (result.next())
31                 {
32                 int x = 1;
33                 Vector v = new Vector();
34                 while (x<=numCol)
35                     {
36                     String col = result.getString(x);
37                     v.add(col);
38                     x++;
39                     }
40                 data.add(v);
41                 }
42
43             stat.close();
44             }
45
46         catch (ArrayIndexOutOfBoundsException ae)
47             {
48             System.out.println("Out of Bounds");
49             }
50
51         catch (SQLException sqle)
52             {
53             sqle.printStackTrace();
54             }
55         catch (Exception e)
56             {
57             e.printStackTrace();
58             }
59
60         return data;
61         }
62 }
```

```
1  //Title:        Definition
2  //Version:      1
3  //Copyright:    Copyright (c) 2001
4  //Author:       Audrey Lecerf
5  //Company:      MAI PROJECT - FUNDP
6  //Description:  Définition de chaque terme du glossaire qui peut comprendre du
                   texte et une illustration
7
8
9  import java.awt.*;
10 import javax.swing.*;
11 import java.util.*;
12 import java.io.*;
13
14 class Definition implements Serializable
15 {
16 String def = null;
17 ImageIcon ill = null;
18
19 public Definition(String d, ImageIcon i)
20     {
21     def = d;
22     ill = i;
23     }
24 }
25
```

```
1  //Title:        Flag
2  //Version:      1
3  //Copyright:    Copyright (c) 2001
4  //Author:       Audrey Lecerf
5  //Company:      MAI PROJECT - FUNDP
6  //Description:  Création de la JComboBox composée de drapeaux et langues
7
8
9  import java.awt.*;
10 import java.awt.event.*;
11
12 import javax.swing.*;
13 import javax.swing.event.*;
14
15 import java.lang.Object.*;
16
17 import java.util.*;
18
19 public class Flag
20 {
21 /* Chargement des icônes */
22
23 ImageIcon ndlflag = new ImageIcon(Student.class.getResource("hollande.gif"));
24 ImageIcon englflag = new ImageIcon(Student.class.getResource("uk.gif"));
25 ImageIcon gerflag = new ImageIcon(Student.class.getResource("germany.gif"));
26 ImageIcon spanflag = new ImageIcon(Student.class.getResource("spain.gif"));
27
28 JComboBox f;
29
30 public Flag ()
31     {
32     /* Attribution des descriptions aux icônes */
33
34     ndlflag.setDescription("Néerlandais");
35     englflag.setDescription("Anglais");
36     gerflag.setDescription("Allemand");
37     spanflag.setDescription("Espagnol");
38
39     /* Création de la JComboBox */
40
41     ImageIcon[] flaglist = {englflag, ndlflag, gerflag, spanflag};
42     f = new JComboBox(flaglist);
43     ComboBoxRenderer renderer= new ComboBoxRenderer();
44     f.setRenderer(renderer);
45     }
46 }
47
```

```
1  //Title:        ImportGlossary
2  //Version:      1
3  //Copyright:    Copyright (c) 2001
4  //Author:       Audrey Lecerf
5  //Company:      MAI PROJECT - FUNDP
6  //Description:  Classe qui propose au professeur de charger un glossaire parmi
                   ceux qui existent
7
9
9  import javax.swing.*;
10 import java.awt.*;
11 import java.awt.event.*;
12 import javax.swing.*;
13 import java.io.*;
14 import java.rmi.*;
15 import java.util.*;
16
17 public class ImportGlossary extends JFrame implements ActionListener
18 {
19 JFrame ig = new JFrame();
20
21 JLabel lup = new JLabel(" ", JLabel.CENTER);
22 JLabel l = new JLabel("Sélectionnez le cours dont vous désirez charger le
   glossaire : ", JLabel.CENTER);
23 JLabel ldown = new JLabel(" ", JLabel.CENTER);
24 Vector vcours = new Vector();
25 noeditjtable courslist;
26 JScrollPane scroll;
27 JButton ok = new JButton("Charger");
28 JButton ko = new JButton("Annuler");
29
30 Teacher teacher;
31 ServerRemote server;
32
33 public ImportGlossary(Teacher tch, ServerRemote s)
34     {
35     teacher = tch;
36     server = s;
37
39     JPanel labels = new JPanel(new GridLayout(3,1));
39     labels.add(lup);
40     labels.add(l);
41     labels.add(ldown);
42
43     JPanel buttons = new JPanel();
44     buttons.add(ok);
45     buttons.add(ko);
46     ok.addActionListener(this);
47     ko.addActionListener(this);
48
49     Vector intitules = new Vector();
50     intitules.add("Titre du cours");
51     intitules.add("Identifiant du cours");
52
53     try
54         {
55         vcours = (Vector)server.loadCours();
56         }
57
58     catch (RemoteException re)
59         {
60         re.printStackTrace();
61         }
62
63     courslist = new noeditjtable(vcours, intitules);
64     scroll = new JScrollPane(courslist);
65     scroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
66
       scroll.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS)
       ;
67
68     ig.getContentPane().setLayout(new BorderLayout());
69     ig.getContentPane().add("North", labels);
70     ig.getContentPane().add("Center", scroll);
71     ig.getContentPane().add("South", buttons);
72
```

```
73      Toolkit t = Toolkit.getDefaultToolkit();
74      Dimension d = t.getScreenSize();
75      int h = d.height;
76      int w = d.width;
77      ig.setSize(500,300);
78      ig.setLocation((w-500)/2 + 100, (h-300)/2 - 50);
79      ig.setResizable(false);
80      ig.show();
81      }
82
83  public void actionPerformed(ActionEvent e)
84      {
85      Object source = e.getSource();
86      if (source == ko)
87         {
88         ig.dispose();
89         }
90      else if (source == ok)
91          {
92          int selection = courslist.getSelectedRow();
93          String coursid = (String)courslist.getValueAt(selection, 1);
94          teacher.remove(teacher.panel);
95          teacher.setVisible(false);
96          ig.dispose();
97          try
98              {
99              // Attention : ne pas changer teacher.cours !!!
100             Vector vtemp = new Vector();
101             vtemp = (Vector)server.loadTermNames(coursid);
102             server.importGlossary(vtemp, coursid, teacher.cours);
103             teacher.vterms = vtemp;
104             new Msg("Glossaire chargé");
105             }
106         catch (RemoteException re)
107             {
108             re.printStackTrace();
109             }
110         teacher.paintInterface();
111         teacher.setVisible(true);
112         teacher.validate();
113         teacher.repaint();
114         }
115     }
116
117 }
```

```
1  //Title:        MAI_Glossary
2  //Version:      1
3  //Copyright:    Copyright (c) 2001
4  //Author:       Audrey Lecerf
5  //Company:      MAI PROJECT - FUNDP
6  //Description:  Lancement du glossaire côté client
7
8
9  import javax.swing.*;
10
11 public class MAI_Glossary
12 {
13 ClientLocal c;
14 public JPanel panel = new JPanel();
15
16 public MAI_Glossary(String categorie, String cours)
17     {
18     c = new ClientLocal(categorie, cours);
19     panel = c.panel;
20     }
21 }
```

```
1  //Title:        Msg
2  //Version:      1
3  //Copyright:    Copyright (c) 2001
4  //Author:       Audrey Lecerf
5  //Company:      MAI PROJECT - FUNDP
6  //Description:  Classe qui affiche un avertissement à l'écran
7
8
9  import javax.swing.*;
10 import java.awt.*;
11 import java.awt.event.*;
12 import javax.swing.*;
13 import java.io.*;
14
15 public class Msg extends JFrame implements ActionListener
16 {
17 JFrame msg = new JFrame();
18 JLabel l = new JLabel();
19 JButton b = new JButton("OK");
20 public Msg(String s)
21     {
22     l.setText(s);
23     msg.getContentPane().setLayout(new BorderLayout());
24     JPanel p = new JPanel();
25     p.add(l, JLabel.CENTER);
26
27     msg.getContentPane().add("Center", p);
28
29     msg.getContentPane().add("South", b);
30     b.addActionListener(this);
31
32     Toolkit t = Toolkit.getDefaultToolkit();
33     Dimension d = t.getScreenSize();
34     int h = d.height;
35     int w = d.width;
36     msg.setSize(500,100);
37     msg.setLocation((w-500)/2, (h-100)/2);
38     msg.setResizable(false);
39     msg.show();
40     }
41
42 public void actionPerformed(ActionEvent e)
43     {
44     Object source = e.getSource();
45     if (source == b)
46        {msg.dispose();
47        }
48     }
49
50 }
```

```
1  //Title:        MsgDelete
2  //Version:      1
3  //Copyright:    Copyright (c) 2001
4  //Author:       Audrey Lecerf
5  //Company:      MAI PROJECT - FUNDP
6  //Description:  Message de confirmation de suppression d'un terme et appel de la
                   méthode distante de suppression de tous les termes appartenant
                   au cours spécifié
7
9
9  import javax.swing.*;
10 import java.awt.*;
11 import java.awt.event.*;
12 import java.io.*;
13 import java.util.*;
14 import javax.swing.border.*;
15 import java.rmi.*;
16 import java.io.*;
17 import javax.swing.text.*;
18
19 public class MsgDelete extends JFrame implements ActionListener
20 {
21 JFrame msg = new JFrame();
22 JPanel buttons = new JPanel();
23 JButton oui = new JButton("Oui");
24 JButton non = new JButton("Non");
25 JLabel l = new JLabel();
26
27 Teacher t;
28 ServerRemote serv;
29 String termname;
30 String cours;
31
32 public MsgDelete(Teacher tch, ServerRemote s, String term)
33     {
34     serv = s;
35     t = tch;
36     termname = term;
37     cours = t.cours;
38
39     msg.getContentPane().setLayout(new BorderLayout());
40     buttons.add(oui);
41     buttons.add(non);
42     oui.addActionListener(this);
43     non.addActionListener(this);
44
45     l.setText("Etes-vous certain de vouloir supprimer le terme : " + termname +
       " ?");
46     l.setHorizontalAlignment(JLabel.CENTER);
47
48     msg.getContentPane().add("South", buttons);
49     msg.getContentPane().add("Center", l);
50
51     Toolkit t = Toolkit.getDefaultToolkit();
52     Dimension d = t.getScreenSize();
53     int h = d.height;
54     int w = d.width;
55     msg.setSize(600,100);
56     msg.setLocation((w-600)/2, (h-100)/2);
57     msg.setResizable(false);
58     msg.show();
59     }
60
61 public void actionPerformed(ActionEvent e)
62     {
63     Object source = e.getSource();
64     if (source == non)
65        {
66         t.newterm.setText("Nouveau terme");
67         t.newabb.setText("Abréviation du nouveau terme");
68         t.newtraducs[0] = "";
69         t.newtraducs[1] = "";
70         t.newtraducs[2] = "";
71         t.newtraducs[3] = "";
72         t.flag.setSelectedIndex(0);
73         t.newtraduc.setText("Nouvelle traduction");
```

```
74          t.newapp.setText("Nouvel apparenté");
75          t.newant.setText("Nouvel opposé");
76          t.appvector.clear();
77          t.antvector.clear();
78          t.uasrefvector.clear();
79          t.deftxt.setText("");
80          t.doc = (Document)(t.kit.createDefaultDocument());
81          t.defimg.setDocument(t.doc);
82          t.imageicon = null;
83          t.repaint();
84          t.revalidate();
85          msg.dispose();
86         }
87      else if (source == oui)
88         {
89         try
90             {
91             serv.deleteTerm(termname, cours);
92             new Msg("Le terme : " + termname + " a été correctement supprimé");
93
94             t.centerpanel.remove(t.termscroll);
95             Vector temp = t.vterms;
96             t.vterms.clear();
97
98             t.uasexpanel.remove(t.uasexscroll);
99             Vector tempuas = t.uasexvector;
100            t.uasexvector.clear();
101
102            try
103                {
104                t.vterms = (Vector)serv.loadTermNames(cours);
105                t.uasexvector = (Vector)serv.loaduaNV(cours);
106                }
107            catch (RemoteException re)
108                {
109                re.printStackTrace();
110                t.vterms = temp;
111                t.uasexvector = tempuas;
112                }
113
114            t.termlist = new JList(t.vterms);
115            t.termscroll = new JScrollPane(t.termlist);
116
               t.termscroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBA
               R_ALWAYS);
117
               t.termscroll.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCRO
               LLBAR_ALWAYS);
118            t.constraints.gridx = 0;
119            t.constraints.gridy = 2;
120            t.constraints.gridheight = 14;
121            t.constraints.gridwidth = 1;
122            t.constraints.weighty = 100;
123            t.layout.setConstraints(t.termscroll, t.constraints);
124            t.centerpanel.add(t.termscroll);
125
126            t.uasexlist = new noeditjtable(t.uasexvector, t.uasextitle);
127            t.uasexlist.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
128            t.uasexscroll = new JScrollPane(t.uasexlist);
129            t.uasexpanel.add(t.uasexscroll);
130
131            t.newterm.setText("Nouveau terme");
132            t.newabb.setText("Abréviation du nouveau terme");
133            t.newtraducs[0] = "";
134            t.newtraducs[1] = "";
135            t.newtraducs[2] = "";
136            t.newtraducs[3] = "";
137            t.flag.setSelectedIndex(0);
138            t.newtraduc.setText("Nouvelle traduction");
139            t.newapp.setText("Nouvel apparenté");
140            t.newant.setText("Nouvel opposé");
141            t.appvector.clear();
142            t.antvector.clear();
143
144            t.uasrefpanel.remove(t.uasrefscroll);
145            t.uasrefvector.clear();
```

```
146             t.uasreflist = new noeditjtable(t.uasrefvector, t.uasreftitle);
147             t.uasreflist.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
148             t.uasrefscroll = new JScrollPane(t.uasreflist);
149             t.uasrefpanel.add(t.uasrefscroll);
150
151             t.deftxt.setText("");
152             t.doc = (Document)(t.kit.createDefaultDocument());
153             t.defimg.setDocument(t.doc);
154             t.imageicon = null;
155             t.repaint();
156             t.revalidate();
157             msg.dispose();
158             }
159         catch (RemoteException re)
160             {
161              re.printStackTrace();
162             }
163         }
164     }
165
166 }
```

```
1  //Title:        MsgDeleteGlossary
2  //Version:      1
3  //Copyright:    Copyright (c) 2001
4  //Author:       Audrey Lecerf
5  //Company:      MAI PROJECT - FUNDP
6  //Description:  Message de confirmation de suppression du glossaire et appel de
                   la méthode distante de suppression d'un terme
7
8
9  import javax.swing.*;
10 import java.awt.*;
11 import java.awt.event.*;
12 import java.io.*;
13 import java.util.*;
14 import javax.swing.border.*;
15 import java.rmi.*;
16 import java.io.*;
17 import javax.swing.text.*;
18
19 public class MsgDeleteGlossary extends JFrame implements ActionListener
20 {
21 JFrame msg = new JFrame();
22 JPanel buttons = new JPanel();
23 JButton oui = new JButton("Oui");
24 JButton non = new JButton("Non");
25 JLabel l = new JLabel();
26
27 Teacher t;
28 ServerRemote serv;
29 String cours;
30
31 public MsgDeleteGlossary(Teacher tch, ServerRemote s)
32     {
33     serv = s;
34     t = tch;
35     cours = t.cours;
36
37     msg.getContentPane().setLayout(new BorderLayout());
38     buttons.add(oui);
39     buttons.add(non);
40     oui.addActionListener(this);
41     non.addActionListener(this);
42
43     l.setText("Etes-vous certain de vouloir supprimer tout le glossaire ? Tous
       les termes y figurant seront effacés à jamais !");
44     l.setHorizontalAlignment(JLabel.CENTER);
45
46     msg.getContentPane().add("South", buttons);
47     msg.getContentPane().add("Center", l);
48
49     Toolkit t = Toolkit.getDefaultToolkit();
50     Dimension d = t.getScreenSize();
51     int h = d.height;
52     int w = d.width;
53     msg.setSize(800,100);
54     msg.setLocation((w-800)/2, (h-100)/2);
55     msg.setResizable(false);
56     msg.show();
57     }
58
59 public void actionPerformed(ActionEvent e)
60     {
61     Object source = e.getSource();
62     if (source == non)
63        {
64         t.newterm.setText("Nouveau terme");
65         t.newabb.setText("Abréviation du nouveau terme");
66         t.newtraducs[0] = "";
67         t.newtraducs[1] = "";
68         t.newtraducs[2] = "";
69         t.newtraducs[3] = "";
70         t.flag.setSelectedIndex(0);
71         t.newtraduc.setText("Nouvelle traduction");
72         t.newapp.setText("Nouvel apparenté");
73         t.newant.setText("Nouvel opposé");
74         t.appvector.clear();
```

```
75          t.antvector.clear();
76          t.uasrefvector.clear();
77
78          t.deftxt.setText("");
79          t.doc = (Document)(t.kit.createDefaultDocument());
80          t.defimg.setDocument(t.doc);
81          t.imageicon = null;
82          t.repaint();
83          t.revalidate();
84          msg.dispose();
85         }
86      else if (source == oui)
87         {
88         try
89             {
90             serv.deleteGlossary(cours);
91             new Msg("Le glossaire a été correctement supprimé");
92             t.centerpanel.remove(t.termscroll);
93             t.uasexpanel.remove(t.uasexscroll);
94             t.vterms.clear();
95             t.vterms = (Vector)serv.loadTermNames(cours);
96             t.uasexvector.clear();
97             t.uasexvector = (Vector)serv.loaduaNV(cours);
98
99             t.termlist = new JList(t.vterms);
100            t.termscroll = new JScrollPane(t.termlist);
101
102            t.uasexlist = new noeditjtable(t.uasexvector, t.uasextitle);
103            t.uasexlist.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
104            t.uasexscroll = new JScrollPane(t.uasexlist);
105            t.uasexpanel.add(t.uasexscroll);
106
107            t.constraints.gridx = 0;
108            t.constraints.gridy = 2;
109            t.constraints.gridheight = 14;
110            t.constraints.gridwidth = 1;
111            t.constraints.weighty = 100;
112            t.layout.setConstraints(t.termscroll, t.constraints);
113            t.centerpanel.add(t.termscroll);
114            t.newterm.setText("Nouveau terme");
115            t.newabb.setText("Abréviation du nouveau terme");
116            t.newtraducs[0] = "";
117            t.newtraducs[1] = "";
118            t.newtraducs[2] = "";
119            t.newtraducs[3] = "";
120            t.flag.setSelectedIndex(0);
121            t.newtraduc.setText("Nouvelle traduction");
122            t.newapp.setText("Nouvel apparenté");
123            t.newant.setText("Nouvel opposé");
124            t.appvector.clear();
125            t.antvector.clear();
126
127            t.uasrefpanel.remove(t.uasrefscroll);
128            t.uasrefvector.clear();
129            t.uasreflist = new noeditjtable(t.uasrefvector, t.uasreftitle);
130            t.uasreflist.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
131            t.uasrefscroll = new JScrollPane(t.uasreflist);
132            t.uasrefpanel.add(t.uasrefscroll);
133
134            t.deftxt.setText("");
135                t.doc = (Document)(t.kit.createDefaultDocument());
136            t.defimg.setDocument(t.doc);
137            t.imageicon = null;
138            t.repaint();
139            t.revalidate();
140            msg.dispose();
141            }
142        catch (RemoteException re)
143            {
144             re.printStackTrace();
145            }
146        }
147     }
148
149 }
```

```
1  //Title:        MsgTerm
2  //Version:      1
3  //Copyright:    Copyright (c) 2001
4  //Author:       Audrey Lecerf
5  //Company:      MAI PROJECT - FUNDP
6  //Description:  Message de confirmation de création d'un terme et appel à la
                   méthode distante de création du terme dans la BD
7
8
9  import javax.swing.*;
10 import java.awt.*;
11 import java.awt.event.*;
12 import java.io.*;
13 import java.util.*;
14 import javax.swing.border.*;
15 import java.rmi.*;
16
17 import java.io.*;
18 import javax.swing.text.*;
19
20 public class MsgTerm extends JFrame implements ActionListener
21 {
22 ServerRemote serv;
23 Term newterm;
24 Teacher t;
25 String cours;
26 JFrame msg = new JFrame();
27 JLabel l1 = new JLabel("      ");
28 JLabel l2 = new JLabel("Etes-vous certain de vouloir créer le terme suivant ? " ,
   JLabel.CENTER);
29 JLabel l3 = new JLabel("      ");
30 JPanel l = new JPanel();
31 JPanel pnord = new JPanel();
32 JPanel psud = new JPanel();
33 JPanel pcenter = new JPanel();
34 JPanel p = new JPanel();
35 JButton oui = new JButton("Oui");
36 JButton non = new JButton("Non");
37 JPanel buttons = new JPanel();
38
39 String term;
40 Definition def;
41 String entrad;
42 String dtrad;
43 String gtrad;
44 String estrad;
45 String newabb;
46 Vector newapp = new Vector();
47 Vector newant = new Vector();
48 Vector newuas = new Vector();
49
50 public MsgTerm(Teacher tch, ServerRemote s, String term, Definition def, String
                  entrad, String dtrad, String gtrad, String estrad, String newabb,
                  Vector newapp, Vector newant, Vector newuas)
51     {
52     serv = s;
53     t = tch;
54     cours = t.cours;
55     this.term = term;
56     this.def = def;
57     this.entrad = entrad;
58     this.dtrad = dtrad;
59     this.gtrad = gtrad;
60     this.estrad = estrad;
61     this.newabb = newabb;
62     this.newapp = newapp;
63     this.newant = newant;
64     this.newuas = newuas;
65
66     msg.getContentPane().setLayout(new BorderLayout());
67     l2.setFont(new java.awt.Font("Dialog", 1, 20));
68     buttons.add(oui);
69     buttons.add(non);
70     oui.addActionListener(this);
71     non.addActionListener(this);
72
```

```
73      pnord.setLayout(new GridLayout(3,2,5,5));
74      psud.setLayout(new GridLayout(1,3,5,5));
75      pcenter.setLayout(new FlowLayout(FlowLayout.CENTER,5,5));
76
77      /* Panel du dessus avec tous les strings */
79
79      JPanel pnom = new JPanel(new GridLayout(1,2));
80      JLabel nom = new JLabel(" Terme : ");
81      JTextField tnom = new JTextField(term);
82      pnom.add(nom);
83      pnom.add(tnom);
84
85      JPanel pabb = new JPanel(new GridLayout(1,2));
86      JLabel abb = new JLabel(" Abréviation : ");
87      JTextField tabb = new JTextField(newabb);
88      pabb.add(abb);
89      pabb.add(tabb);
90
91      JPanel pen = new JPanel(new GridLayout(1,2));
92      JLabel en = new JLabel(" Traduction anglaise : ");
93      JTextField ten = new JTextField(entrad);
94      pen.add(en);
95      pen.add(ten);
96
97      JPanel pdu = new JPanel(new GridLayout(1,2));
98      JLabel du = new JLabel(" Traduction néerlandaise : ");
99      JTextField tdu = new JTextField(dtrad);
100     pdu.add(du);
101     pdu.add(tdu);
102
103     JPanel pge = new JPanel(new GridLayout(1,2));
104     JLabel ge = new JLabel(" Traduction allemande : ");
105     JTextField tge = new JTextField(gtrad);
106     pge.add(ge);
107     pge.add(tge);
108
109     JPanel pes = new JPanel(new GridLayout(1,2));
110     JLabel es = new JLabel(" Traduction espagnole : ");
111     JTextField tes = new JTextField(estrad);
112     pes.add(es);
113     pes.add(tes);
114
115     pnom.setBorder(BorderFactory.createLineBorder(Color.gray));
116     pabb.setBorder(BorderFactory.createLineBorder(Color.gray));
117     pen.setBorder(BorderFactory.createLineBorder(Color.gray));
118     pdu.setBorder(BorderFactory.createLineBorder(Color.gray));
119     pge.setBorder(BorderFactory.createLineBorder(Color.gray));
120     pes.setBorder(BorderFactory.createLineBorder(Color.gray));
121
122     pnord.add(pnom);
123     pnord.add(pabb);
124     pnord.add(pen);
125     pnord.add(pdu);
126     pnord.add(pge);
127     pnord.add(pes);
128
129     /* Panel du milieu avec la définition */
130
131     JTextPane defarea = new JTextPane();
132     StyledEditorKit kit = new StyledEditorKit(); ;
133     Document doc = (Document)(kit.createDefaultDocument());
134     defarea.setEditorKit(kit);
135     defarea.setDocument(doc);
136
137     if (def.ill != null)
138         {
139         defarea.insertIcon(def.ill);
140         defarea.setText("  " + def.def);
141         }
142
143     else defarea.setText(def.def);
144
145     defarea.setEditable(false);
146     JScrollPane defscroll = new JScrollPane(defarea);
147
        defscroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDE
```

```
        D);
148
        defscroll.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_N
        EEDED);
149     defscroll.setBorder(BorderFactory.createTitledBorder("Définition"));
150     defscroll.setPreferredSize(new Dimension(610, 180));
151     pcenter.add(defscroll);
152
153     /* Panel du dessous avec les vecteurs transformés en JTables */
154
155     /* Termes apparentés */
156
157     JPanel app = new JPanel();
158     app.setLayout(new BorderLayout());
159     JLabel applabel = new JLabel("Termes apparentés", JLabel.CENTER);
160     JList applist = new JList(newapp);
161     JScrollPane appscroll = new JScrollPane(applist);
162     app.add("North", applabel);
163     app.add("Center", appscroll);
164
165     /* Termes opposés */
166
167     JPanel ant = new JPanel();
168     ant.setLayout(new BorderLayout());
169     JLabel antlabel = new JLabel("Termes opposés", JLabel.CENTER);
170     JList antlist = new JList(newant);
171     JScrollPane antscroll = new JScrollPane(antlist);
172     ant.add("North", antlabel);
173     ant.add("Center", antscroll);
174
175     /* UAS référencées */
176
177     JPanel uas = new JPanel();
178     uas.setLayout(new BorderLayout());
179     JLabel uaslabel = new JLabel("UAS référencées", JLabel.CENTER);
180     Vector uastitle = new Vector();
181     uastitle.add("Titre");
182     uastitle.add("Version");
183     Vector uasvector = new Vector();
184
185     for (int x=0 ; x<newuas.size() ; x++)
186         {
187         Vector vtemp = new Vector();
188         vtemp.add((String)((UA)newuas.get(x)).titre);
189         vtemp.add((String)Integer.toString((int)((UA)newuas.get(x)).version));
190         uasvector.add(vtemp);
191         }
192
193     noeditjtable uaslist = new noeditjtable(uasvector, uastitle);
194     JScrollPane uasscroll = new JScrollPane(uaslist);
195     uas.add("North", uaslabel);
196     uas.add("Center", uasscroll);
197
198     psud.add(app);
199     psud.add(ant);
200     psud.add(uas);
201
202     /* Mise en forme */
203
204     l.setLayout(new GridLayout(3,1));
205     l.add(l1);
206     l.add(l2);
207     l.add(l3);
208
209     p.setLayout(new GridLayout(3,1));
210     p.add(pnord);
211     p.add(pcenter);
212     p.add(psud);
213
214     msg.getContentPane().add("North", l);
215     msg.getContentPane().add("South", buttons);
216     msg.getContentPane().add("Center", p);
217
218     Toolkit t = Toolkit.getDefaultToolkit();
219     Dimension d = t.getScreenSize();
220     int h = d.height;
```

```
221       int w = d.width;
222       msg.setSize(620,700);
223       msg.setLocation((w-620)/2, (h-700)/2);
224       msg.setResizable(false);
225       msg.show();
226       }
227
228   public void actionPerformed(ActionEvent e)
229       {
230       Object source = e.getSource();
231       if (source == non)
232          {
233          msg.dispose();
234          }
235       else if (source == oui)
236          {
237          try
238              {
239              newterm = serv.saveTerm(term, newabb, def, entrad, dtrad, gtrad,
                 estrad, newapp, newant, newuas, cours);
240
241              t.centerpanel.remove(t.termscroll);
242              t.uasexpanel.remove(t.uasexscroll);
243              Vector temp = t.vterms;
244              Vector tempuas = t.uasexvector;
245              t.vterms.clear();
246              t.uasexvector.clear();
247              try
248                  {
249                  t.vterms = (Vector)serv.loadTermNames(cours);
250                  t.uasexvector = (Vector)serv.loaduaNV(cours);
251                  }
252              catch (RemoteException re)
253                  {
254                  re.printStackTrace();
255                  t.vterms = temp;
256                  t.uasexvector = tempuas;
257                  }
258
259              t.termlist = new JList(t.vterms);
260              t.termscroll = new JScrollPane(t.termlist);
261
                 t.termscroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBA
                 R_ALWAYS);
262
                 t.termscroll.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCRO
                 LLBAR_ALWAYS);
263              t.constraints.gridx = 0;
264              t.constraints.gridy = 2;
265              t.constraints.gridheight = 14;
266              t.constraints.gridwidth = 1;
267              t.constraints.weighty = 100;
268              t.layout.setConstraints(t.termscroll, t.constraints);
269              t.centerpanel.add(t.termscroll);
270
271              t.uasexlist = new noeditjtable(t.uasexvector, t.uasextitle);
272              t.uasexlist.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
273              t.uasexscroll = new JScrollPane(t.uasexlist);
274              t.uasexpanel.add(t.uasexscroll);
275
276              t.newterm.setText("Nouveau terme");
277              t.newabb.setText("Abréviation du nouveau terme");
278              t.newtraducs[0] = "";
279              t.newtraducs[1] = "";
280              t.newtraducs[2] = "";
281              t.newtraducs[3] = "";
282              t.flag.setSelectedIndex(0);
283              t.newtraduc.setText("Nouvelle traduction");
284              t.newapp.setText("Nouvel apparenté");
285              t.newant.setText("Nouvel opposé");
286              t.appvector.clear();
287              t.antvector.clear();
288
289              t.uasrefpanel.remove(t.uasrefscroll);
290              t.uasrefvector.clear();
291              t.uasreflist = new noeditjtable(t.uasrefvector, t.uasreftitle);
```

```
292             t.uasreflist.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
293             t.uasrefscroll = new JScrollPane(t.uasreflist);
294             t.uasrefpanel.add(t.uasrefscroll);
295
296              t.deftxt.setText("");
297             t.doc = (Document)(t.kit.createDefaultDocument());
298             t.defimg.setDocument(t.doc);
299             t.imageicon = null;
300
301             t.repaint();
302             t.revalidate();
303             msg.dispose();
304             new Msg("Le terme : " + newterm.term + " a été correctement
                enregistré");
305             }
306
307         catch (RemoteException re)
308             {
309              re.printStackTrace();
310             }
311
312         }
313     }
314
315 }
```

```
1  //Title:        noeditjtable
2  //Version:      1
3  //Copyright:    Copyright (c) 2001
4  //Author:       Audrey Lecerf
5  //Company:      MAI PROJECT - FUNDP
6  //Description:  JTable personnalisée à une utilisation de simple consultation
7
8
9  import javax.swing.*;
10 import javax.swing.border.*;
11 import java.awt.*;
12 import java.awt.event.*;
13 import java.util.*;
14 import java.io.*;
15 import java.sql.*;
16
17 public  class noeditjtable extends JTable
18 {
19
20 public noeditjtable()
21     {
22     super();
23     DefaultListSelectionModel dlsm = new DefaultListSelectionModel();
24     setSelectionMode(dlsm.SINGLE_SELECTION);
25     }
26
27 public noeditjtable(Vector x, Vector y)
28     {
29     super(x,y);
30     DefaultListSelectionModel dlsm = new DefaultListSelectionModel();
31     setSelectionMode(dlsm.SINGLE_SELECTION);
32     }
33
34 public boolean isCellEditable (int x,int y)
35     {
36     return (false);
37     }
38
39 }
```

```
1  //Title:        PagesLoading
2  //Version:      1
3  //Copyright:    Copyright (c) 2001
4  //Author:       Audrey Lecerf
5  //Company:      MAI PROJECT - FUNDP
6  //Description:  Chargement des pages de l'UA identifiée par le String passé en
                   argument (parsing nécessaire)
7
8
9  import java.awt.*;
10 import java.lang.Object.*;
11 import javax.swing.*;
12 import java.awt.event.*;
13 import java.util.*;
14 import java.sql.*;
15 import java.sql.Connection.*;
16 import java.rmi.*;
17 import java.io.*;
18
19 import javax.swing.event.*;
20 import javax.swing.border.*;
21
22 public class PagesLoading implements Serializable
23 {
24 UA ua;
25 PageUA page;
26 String titre;
27 int version;
28 Connection con;
29 ServerRemote server;
30 Vector pages = new Vector();
31
32 public PagesLoading(String t, int v, ServerRemote s) throws RemoteException
33     {
34     server = s;
35     titre = t;
36     version = v;
37
38     try
39         {
40         Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
41         con = DriverManager.getConnection("jdbc:odbc:bdmai", "sa", "xxxxxx");
42         DatabaseMetaData dma = con.getMetaData();
43         }
44
45     catch (Exception e)
46         {
47         e.printStackTrace();
48         }
49
50     loadPages();
51
52     try
53         {
54         con.close();
55         }
56
57     catch(Exception e)
58         {
59         e.printStackTrace();
60         }
61
62     }
63
64 public void loadPages() throws RemoteException
65     {
66     try
67             {
68             Vector v = new Vector();
69             v = ((DB)server.giveDB()).getData("SELECT * FROM UA WHERE TITRE = '"
               + titre + "' AND VERSION = " + version, con);
70
71             int sizev = v.size();
72             int x = 0;
73
74             while (x < sizev)
```

```
75                  {
76                  String titre = null;
77                  int version;
78                  String presentation = null;
79                  String statut = null;
80                  String faculte = null;
81
82                  titre = (String)((Vector)(v.get(x))).get(0);
83
84                  presentation = (String)((Vector)(v.get(x))).get(1);
85
86                  statut = (String)((Vector)(v.get(x))).get(3);
87
88                  version =
                    (int)Integer.parseInt((String)((Vector)(v.get(x))).get(4));
89
90                  faculte = (String)((Vector)(v.get(x))).get(5);
91
92                  ua = new UA(titre, presentation, statut, version, faculte);
93
94                  x++;
95                  }
96
97              Vector p = new Vector();
98              p = ((DB)server.giveDB()).getData("SELECT * FROM PAGE WHERE TITRE_UA
                = '" + titre + "' AND VERSION_UA = " + version, con);
99
100             int sizep = p.size();
101             x = 0;
102
103             while (x < sizep)
104                 {
105                 int num;
106                 String cont = null;
107                 String titr = null;
108
109                 titr = (String)((Vector)(p.get(x))).get(2);
110                 num =
                    (int)Integer.parseInt((String)((Vector)(p.get(x))).get(3));
111                 cont = (String)((Vector)(p.get(x))).get(4);
112                 page = new PageUA(ua, titr, num, cont);
113
114                 pages.add(page);
115
116                 x++;
117                 }
118             }
119
120         catch (Exception e)
121             {
122             e.printStackTrace();
123             }
124     }
125 }
126
127
```

```
1  //Title:        PageUA
2  //Version:      1
3  //Copyright:    Copyright (c) 2001
4  //Author:       Audrey Lecerf
5  //Company:      MAI PROJECT - FUNDP
6  //Description:  Création d'une page
7
8  import java.io.*;
9
10 public class PageUA implements Serializable
11 {
12 UA ua;
13 int num;
14 String cont;
15 String titre;
16
17 public PageUA(UA ua, String titre, int num, String cont)
18     {
19     this.ua = ua;
20     this.num = num;
21     this.cont = cont;
22     this.titre = titre;
23     }
24 }
25
```

```
1  //Title:        SaveTerm
2  //Version:      1
3  //Copyright:    Copyright (c) 2001
4  //Author:       Audrey Lecerf
5  //Company:      MAI PROJECT - FUNDP
6  //Description:  Création du nouveau terme et insertion dans la base de données
7  //                                        ------------------------------------
8
9  import java.awt.*;
10 import java.awt.event.*;
11
12 import javax.swing.*;
13 import javax.swing.event.*;
14
15 import java.util.*;
16 import java.io.*;
17
18 import java.sql.*;
19 import java.sql.Connection.*;
20
21 import java.rmi.*;
22
23 public class SaveTerm implements Serializable
24 {
25 String newterm;
26 String newabb;
27 Definition newdef;
28 String cours;
29 String entrad;
30 String dtrad;
31 String gtrad;
32 String estrad;
33 Vector newapp;
34 Vector newant;
35 Vector newuas;
36 String nomimage = "";
37 Connection con;
38 Ressource_Remote obj;
39
40 public SaveTerm(Term t)
41     {
42     cours = t.cours;
43     newterm = t.term;
44     newabb = t.abbreviation;
45     newdef = t.definition;
46     entrad = t.englishtranslation;
47     dtrad = t.dutchtranslation;
48     gtrad = t.germantranslation;
49     estrad = t.spanishtranslation;
50     newapp = t.relatedterms;
51     newant = t.antonyms;
52     newuas = t.uas;
53
54     try
55         {
56         Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
57         con = DriverManager.getConnection("jdbc:odbc:bdmai", "sa", "xxxxxx");
58         DatabaseMetaData dma = con.getMetaData();
59         }
60
61     catch (Exception e)
62         {
63         e.printStackTrace();
64         }
65
66     try
67         {
68         obj = (Ressource_Remote)
           Naming.lookup("//138.48.32.193/servlets/Ressource_Implementation");
69         }
70
71     catch(Exception e)
72         {
73         e.printStackTrace();
74         obj = null;
75         }
```

```
76
77
78      saveTermInBD();
79
80      try
81          {
82          con.close();
83          }
84
85      catch(Exception e)
86          {
87          e.printStackTrace();
88          }
89      }
90
91  public void saveTermInBD()
92      {
93      /* Création de l'image dans la bd ==> Ca fonctionne !!! */
94      try
95          {
96          if (newdef.ill != null)
97              {
98              String filepath = (String)newdef.ill.getDescription();
99              File f = new File(filepath);
100             FileInputStream is = new FileInputStream(f);
101             int av = is.available();
102             byte[] b = new byte[av];
103             is.read(b);
104             is.close();
105             int l = (int)f.length();
106
107             int index1 = filepath.lastIndexOf(".");
108             int index2 = filepath.lastIndexOf("\\");
109             String typeimage = filepath.substring(index1+1);
110             nomimage = filepath.substring(index2+1);
111             String nomcourt = filepath.substring(index2+1, index1);
112             Ressource ress = new
                Ressource(888,typeimage,"informatique",nomcourt,"Lecerf","Audrey","0
                7/20/2001", f, b, l);
113             obj.Set_Ressource(ress);
114             }
115         }
116
117     catch (Exception e)
118         {
119         e.printStackTrace();
120         }
121
122     /* Création du terme dans la bd */
123     try
124         {
125             String insertTerm ="INSERT INTO TERM VALUES (?,?,?,?,?,?,?,?,?,?)";
126             PreparedStatement pstT= con.prepareStatement(insertTerm);
127             pstT.setString(1, newterm);
128             pstT.setString(2, cours);
129             pstT.setString(3, newabb);
130             pstT.setString(4, entrad);
131             pstT.setString(5, dtrad);
132             pstT.setString(6, gtrad);
133             pstT.setString(7, estrad);
134             pstT.setString(8, newdef.def);
135             pstT.setString(9, "");
136             pstT.setString(10, nomimage);
137             pstT.executeUpdate();
138
139
140             for (int xapp=0 ; xapp<newapp.size() ; xapp++)
141                 {
142                 String insertRT ="INSERT INTO RELATEDTERM VALUES (?,?,?)";
143                 PreparedStatement pstRT= con.prepareStatement(insertRT);
144                 pstRT.setString(1, newterm);
145                 pstRT.setString(2, cours);
146                 pstRT.setString(3, (String)newapp.get(xapp));
147                 pstRT.executeUpdate();
148                 }
149
```

```
150                 for (int xant=0 ; xant<newant.size() ; xant++)
151                     {
152                     String insertA ="INSERT INTO ANTONYM VALUES (?,?,?)";
153                     PreparedStatement pstA= con.prepareStatement(insertA);
154                     pstA.setString(1, newterm);
155                     pstA.setString(2, cours);
156                     pstA.setString(3, (String)newant.get(xant));
157                     pstA.executeUpdate();
158                     }
159
160                 for (int xuas=0 ; xuas<newuas.size() ; xuas++)
161                     {
162                     String insertC ="INSERT INTO CHARACTERIZATION VALUES
                        (?,?,?,?,?)";
163                     PreparedStatement pstC= con.prepareStatement(insertC);
164                     String titreUA = (String)((UA)newuas.get(xuas)).titre;
165                     String shortnameUA = (String)((UA)newuas.get(xuas)).titre;
166                     int versionUA = (int)((UA)newuas.get(xuas)).version;
167
168                     pstC.setString(1, newterm);
169                     pstC.setString(2, cours);
170                     pstC.setString(3, titreUA);
171                     pstC.setInt(4, versionUA);
172                     pstC.setString(5, shortnameUA);
173                     pstC.executeUpdate();
174                     }
175
176             }
177
178         catch(SQLException exp)
179             {
180             System.out.println("SQLException");
181             exp.printStackTrace();
182             }
183
184         }
185
186     }
```

```
1  //Title:        ServerImpl
2  //Version:      1
3  //Copyright:    Copyright (c) 2001
4  //Author:       Audrey Lecerf
5  //Company:      MAI PROJECT - FUNDP
6  //Description:  Lancement du glossaire côté serveur
7
8
9  import java.rmi.*;
10 import java.rmi.server.*;
11 import java.rmi.registry.*;
12
13 import java.sql.*;
14 import java.sql.Connection.*;
15 import java.util.*;
16
17 import java.net.*;
18
19 public class ServerImpl     extends UnicastRemoteObject
20                             implements ServerRemote
21 {
22 Connection con;
23 DB db;
24
25 public ServerImpl() throws RemoteException
26     {
27     db = new DB();
28
29     try
30         {
31         /* Binding the object ServerImpl to the name "glossary" in the registry
           */
32
33         Registry repertoire;
34         repertoire =
           (Registry)LocateRegistry.getRegistry("milan.info.fundp.ac.be");
35         repertoire.rebind("glossary", this);
36         System.out.println("Server is ready...");
37         }
38
39
40     catch (RemoteException re)
41         {
42         re.printStackTrace();
43         }
44
45     catch (Exception e)
46         {
47         e.printStackTrace();
48         }
49
50     }
51
52 public DB giveDB() throws RemoteException
53     {
54     return db;
55     }
56
57 public Vector loadTermNames(String cours) throws RemoteException
58   {
59   con = null;
60
61     try
62         {
63         /* Connexion à la base de données "bdmai" */
64
65         Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
66         con = DriverManager.getConnection("jdbc:odbc:bdmai", "sa", "xxxxxx");
67         DatabaseMetaData dma = con.getMetaData();
68         }
69
70     catch (Exception e)
71         {
72         e.printStackTrace();
73         }
74
```

```
75    Vector v = new Vector();
76    Vector termsfromcourse = new Vector();
77
78    try
79      {
80      /* Demander la liste des noms des termes existant dans la BD et appartenant
        au cours indiqué */
81
82      v = (Vector)db.getData("select Name from Term where Cours = '" + cours +
        "'", con);
83
84      int sizev = v.size();
85      int x = 0;
86
87      while (x < sizev)
88          {
89          String name = null;
90
91          name = (String)((Vector)(v.get(x))).get(0);
92
93          termsfromcourse.add(name);
94          x++;
95          }
96      }
97
98    catch (Exception e)
99      {
100     e.printStackTrace();
101     }
102
103     try
104         {
105         con.close();
106         }
107
108     catch(Exception e)
109         {
110         e.printStackTrace();
111         }
112
113
114    return termsfromcourse;
115    }
116
117 public Vector loaduaNV(String cours) throws RemoteException
118     {
119     con = null;
120
121     try
122         {
123         /* Connexion à la base de données "bdmai" */
124
125         Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
126         con = DriverManager.getConnection("jdbc:odbc:bdmai", "sa", "xxxxxx");
127         DatabaseMetaData dma = con.getMetaData();
128         }
129
130     catch (Exception e)
131         {
132         e.printStackTrace();
133         }
134
135     Vector v = new Vector();
136     Vector uasfromcourse = new Vector();
137
138     try
139     {
140     /* Demander la liste des noms et versions des UAS existant dans la BD */
141
142     v = (Vector)db.getData("SELECT DISTINCT TITRE, VERSION FROM CHARACTERIZATION
        WHERE COURS = '" + cours + "' UNION SELECT DISTINCT TITRE, VERSION FROM UA
        WHERE COURS = '" + cours + "'", con);
143
144     int sizev = v.size();
145     int x = 0;
146
```

```
147         while (x < sizev)
148             {
149             Vector NV = new Vector();
150             String titre;
151             String version;
152
153             titre = (String)((Vector)(v.get(x))).get(0);
154             version = (String)((Vector)(v.get(x))).get(1);
155
156             NV.add(titre);
157             NV.add(version);
158             uasfromcourse.add(NV);
159             x++;
160             }
161         }
162
163         catch (Exception e)
164             {
165             e.printStackTrace();
166             }
167
168         try
169             {
170             con.close();
171             }
172
173         catch(Exception e)
174             {
175             e.printStackTrace();
176             }
177
178
179         return uasfromcourse;
180         }
181
182     public Vector loadCours() throws RemoteException
183         {
184         con = null;
185
186         try
187             {
188             /* Connexion à la base de données "bdmai" */
189
190             Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
191             con = DriverManager.getConnection("jdbc:odbc:bdmai", "sa", "xxxxxx");
192             DatabaseMetaData dma = con.getMetaData();
193             }
194
195         catch (Exception e)
196             {
197             e.printStackTrace();
198             }
199
200         Vector v = new Vector();
201         Vector cours = new Vector();
202
203         try
204         {
205         /* Demander la liste des cours possédant un glossaire */
206
207         v = (Vector)db.getData("select DISTINCT C.TITRE_COURS, C.ID_FACULTE from
            COURS C, TERM T where C.ID_FACULTE = T.COURS" , con);
208
209         int sizev = v.size();
210         int x = 0;
211
212         while (x < sizev)
213             {
214             String titre;
215             String id;
216             titre = (String)((Vector)(v.get(x))).get(0);
217             id = (String)((Vector)(v.get(x))).get(1);
218             Vector temp = new Vector();
219             temp.add(titre);
220             temp.add(id);
221             cours.add(temp);
```

```
222             x++;
223             }
224         }
225
226         catch (Exception e)
227             {
228             e.printStackTrace();
229             }
230
231         try
232             {
233             con.close();
234             }
235
236         catch(Exception e)
237             {
238             e.printStackTrace();
239             }
240
241         return cours;
242         }
243
244     public Vector loadAPP(String cours) throws RemoteException
245         {
246         con = null;
247
248         try
249             {
250             /* Connexion à la base de données "bdmai" */
251
252             Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
253             con = DriverManager.getConnection("jdbc:odbc:bdmai", "sa", "xxxxxx");
254             DatabaseMetaData dma = con.getMetaData();
255             }
256
257         catch (Exception e)
258             {
259             e.printStackTrace();
260             }
261
262           Vector v = new Vector();
263           Vector app = new Vector();
264
265           try
266             {
267             /* Demander la liste des relatedterms existant dans la BD et appartenant
              au cours indiqué */
268
269             v = (Vector)db.getData("select RT_Name from RelatedTerm where Cours = '"
              + cours + "'", con);
270
271             int sizev = v.size();
272             int x = 0;
273
274             while (x < sizev)
275                 {
276                 String name = null;
277
278                 name = (String)((Vector)(v.get(x))).get(0);
279
280                 app.add(name);
281                 x++;
282                 }
283             }
284
285           catch (Exception e)
286             {
287             e.printStackTrace();
288             }
289
290             try
291             {
292             con.close();
293             }
294
295             catch(Exception e)
```

```
296         {
297         e.printStackTrace();
298         }
299
300
301     return app;
302     }
303
304 public Vector loadANT(String cours) throws RemoteException
305     {
306     con = null;
307
308     try
309         {
310         /* Connexion à la base de données "bdmai" */
311
312         Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
313         con = DriverManager.getConnection("jdbc:odbc:bdmai", "sa", "xxxxxx");
314         DatabaseMetaData dma = con.getMetaData();
315         }
316
317     catch (Exception e)
318         {
319         e.printStackTrace();
320         }
321
322     Vector v = new Vector();
323     Vector ant = new Vector();
324
325     try
326         {
327         /* Demander la liste des antonyms existant dans la BD et appartenant au
            cours indiqué */
328
329         v = (Vector)db.getData("select ANT_Name from Antonym where Cours = '" +
            cours + "'", con);
330
331         int sizev = v.size();
332         int x = 0;
333
334         while (x < sizev)
335             {
336             String name = null;
337
338             name = (String)((Vector)(v.get(x))).get(0);
339
340             ant.add(name);
341             x++;
342             }
343         }
344
345     catch (Exception e)
346         {
347         e.printStackTrace();
348         }
349
350     try
351         {
352         con.close();
353         }
354
355     catch(Exception e)
356         {
357         e.printStackTrace();
358         }
359
360
361    return ant;
362     }
363
364 public Term loadTerm(String termname, String cours) throws RemoteException
365     {
366     TermLoading tl = new TermLoading(termname, cours, this);
367     return (Term)tl.term;
368     }
369
```

```
370  public Vector loadUAS(Vector uasnames, String cours) throws RemoteException
371      {
372      CreateUasVector v = new CreateUasVector(uasnames, this);
373      return (Vector)v.uas;
374      }
375
376  public Vector loadPages(String titre, int version) throws RemoteException
377      {
378      PagesLoading pl = new PagesLoading(titre, version, this);
379      return (Vector)pl.pages;
380      }
381
382  public Term saveTerm(String nt, String abb, Definition def, String entrad,
                         String dtrad, String gtrad, String estrad, Vector
                         appvector, Vector antvector, Vector uasvector, String
                         cours) throws RemoteException
383      {
384      Term t = new Term(nt, def, cours, entrad, dtrad, gtrad, estrad, abb,
         appvector, antvector, uasvector);
385      SaveTerm st = new SaveTerm(t);
386      return t;
387      }
388
389  public void deleteTerm(String name, String cours) throws RemoteException
390      {
391      con = null;
392
393      try
394          {
395          /* Connexion à la base de données "bdmai" */
396
397          Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
398          con = DriverManager.getConnection("jdbc:odbc:bdmai", "sa", "xxxxxx");
399          DatabaseMetaData dma = con.getMetaData();
400          }
401
402      catch (Exception e)
403          {
404          e.printStackTrace();
405          }
406
407      String reqchar = "delete from CHARACTERIZATION where NAME = '" + name + "'
         AND COURS = '" + cours + "'";
408      String reqrt = "delete from RELATEDTERM where NAME = '" + name + "' AND
         COURS = '" + cours + "'";
409      String reqant = "delete from ANTONYM where NAME = '" + name + "' AND COURS =
         '" + cours + "'";
410      String req = "delete from TERM where NAME = '" + name + "' AND COURS = '" +
         cours + "'";
411
412      try
413          {
414          PreparedStatement ps1 = con.prepareStatement(reqchar);
415          ps1.executeUpdate();
416
417          PreparedStatement ps2 = con.prepareStatement(reqrt);
418          ps2.executeUpdate();
419
420          PreparedStatement ps3 = con.prepareStatement(reqant);
421          ps3.executeUpdate();
422
423          PreparedStatement ps5 = con.prepareStatement(req);
424          ps5.executeUpdate();
425          }
426
427      catch (SQLException se)
428          {
429          se.printStackTrace();
430          }
431
432      try
433          {
434          con.close();
435          }
436
437      catch(Exception e)
```

```
438         {
439         e.printStackTrace();
440         }
441
442     }
443
444 public void deleteGlossary(String cours) throws RemoteException
445     {
446     con = null;
447
448     try
449         {
450         /* Connexion à la base de données "bdmai" */
451
452         Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
453         con = DriverManager.getConnection("jdbc:odbc:bdmai", "sa", "xxxxxx");
454         DatabaseMetaData dma = con.getMetaData();
455         }
456
457     catch (Exception e)
458         {
459         e.printStackTrace();
460         }
461
462     String req1 = "delete from RelatedTerm where Cours = '" + cours + "'";
463     String req2 = "delete from Antonym where Cours = '" + cours + "'";
464     String req3 = "delete from Characterization where Cours = '" + cours + "'";
465     String req4 = "delete from Term where Cours = '" + cours + "'";
466
467     try
468         {
469         PreparedStatement ps1 = con.prepareStatement(req1);
470         ps1.executeUpdate();
471
472         PreparedStatement ps2 = con.prepareStatement(req2);
473         ps2.executeUpdate();
474
475         PreparedStatement ps3 = con.prepareStatement(req3);
476         ps3.executeUpdate();
477
478         PreparedStatement ps4 = con.prepareStatement(req4);
479         ps4.executeUpdate();
480         }
481
482     catch (SQLException se)
483         {
484         se.printStackTrace();
485         }
486
487     try
488         {
489         con.close();
490         }
491
492     catch(Exception e)
493         {
494         e.printStackTrace();
495         }
496
497     }
498
499 public void importGlossary(Vector terms, String coursfrom, String coursto)
    throws RemoteException
500     {
501     for (int x=0 ; x<terms.size() ; x++)
502         {
503         /* On ajoute un nouveau terme dont le cours est différent sans modifier
            le terme existant */
504         Term t = loadTerm(changeAp((String)terms.get(x)), coursfrom);
505         t.cours = coursto;
506         SaveTerm st = new SaveTerm(t);
507         }
508     }
509
510 /* Change a string containing a "'" into another string that can be accepted by
    a SQL query */
```

```
511
512  private String changeAp(String are)
513      {
514      String neware = are;
515      int indof = neware.indexOf("'", 0);
516
517      while (indof >=0)
518          {
519          String start = neware.substring(0, indof);
520          String end = neware.substring(indof+1);
521          neware = start + "''" + end;
522          indof = neware.indexOf("'", indof+2);
523          }
524
525      return neware;
526      }
527
528  }
```

```
1  //Title:        ServerRemote
2  //Version:      1
3  //Copyright:    Copyright (c) 2001
4  //Author:       Audrey Lecerf
5  //Company:      MAI PROJECT - FUNDP
6  //Description:  Interface correspondant à la classe ServerImpl
7
8
9  import java.rmi.*;
10 import java.sql.*;
11 import java.util.*;
12
13 public interface ServerRemote extends Remote
14 {
15 DB giveDB() throws RemoteException;
16
17 Vector loadTermNames(String cours) throws RemoteException; // Méthode à utiliser
                                                               avant de parser le
                                                               contenu d'une UA
18 Vector loaduaNV(String cours) throws RemoteException;
19 Term loadTerm(String termname, String cours) throws RemoteException;
20 Vector loadUAS(Vector uasnames, String cours) throws RemoteException;
21 Vector loadPages(String titre, int version) throws RemoteException;
22 Vector loadCours() throws RemoteException;
23 Vector loadAPP(String cours) throws RemoteException;
24 Vector loadANT(String cours) throws RemoteException;
25 void deleteTerm(String termname, String cours) throws RemoteException;
26 void deleteGlossary(String cours) throws RemoteException;
27 void importGlossary(Vector terms, String coursfrom, String coursto)
   throws RemoteException;
28 Term saveTerm(String nt, String abb, Definition def, String entrad, String
                dtrad, String gtrad, String estrad, Vector appvector, Vector
                antvector, Vector uasvector, String cours) throws RemoteException;

29 }
```

```
1  //Title:        ShowUA
2  //Version:      1
3  //Copyright:    Copyright (c) 2001
4  //Author:       Audrey Lecerf
5  //Company:      MAI PROJECT - FUNDP
6  //Description:  Affichage de l'UA sélectionnée dans une fenêtre où l'on peut
                   choisir la page à afficher
7
8
9  import java.awt.*;
10 import javax.swing.*;
11 import java.util.*;
12 import java.io.*;
13 import java.awt.event.*;
14 import javax.swing.event.*;
15 import java.net.URL;
16 import java.net.MalformedURLException;
17 import javax.swing.text.html.*;
18 import javax.swing.text.*;
19
20
21 class ShowUA extends JFrame implements ActionListener
22 {
23 PageUA p;
24 JPanel panel = new JPanel();
25 Choice pagenum;
26 Vector vpages = new Vector();
27 JEditorPane area = new JEditorPane();
28 JButton b = new JButton("Retour au glossaire");
29
30 public ShowUA(Vector pages)
31     {
32     super((String)((UA)((PageUA)pages.get(0)).ua).titre);
33     vpages = pages;
34     int vsize = vpages.size();
35     int x = 0;
36     pagenum = new Choice();
37
38     while (x < vsize)
39         {
40         pagenum.addItem("Page " + x);
41         x++;
42         }
43
44     area.setText("Choisissez la page que vous désirez voir affichée...");
45     area.setEditable(false);
46     JScrollPane scroll = new JScrollPane(area);
47
48     this.setBackground(Color.orange);
49     panel.setBackground(Color.orange);
50     Toolkit t = Toolkit.getDefaultToolkit();
51     Dimension d = t.getScreenSize();
52     int h = d.height;
53     int w = d.width;
54
55     b.setActionCommand("back");
56     b.addActionListener(this);
57
58     pagenum.addItemListener(new ItemListener()
59     {
60     public void itemStateChanged(ItemEvent e)
61         {
62         Choice ch = (Choice)e.getSource();
63         String sel = (String)ch.getSelectedItem().trim();
64         int y = 0;
65         while (y < pagenum.getItemCount() && !sel.equals("Page " + y))
66                 {
67                 y++;
68                 }
69         if (sel.equals("Page " + y))
70                 {
71                 p = (PageUA)vpages.get(y);
72
73                 HTMLEditorKit kit = new HTMLEditorKit();
74                 HTMLDocument doc = (HTMLDocument)(kit.createDefaultDocument());
75
```

```
76                  area.setEditorKit(kit);
77                  area.setDocument(doc);
78                  area.setContentType("text/html");
79                  area.setText(p.cont);
80                  }
81          }
82      });
83
84      panel.add(pagenum);
85      panel.add(b);
86
87      this.getContentPane().setLayout(new BorderLayout());
88      this.getContentPane().add(scroll, BorderLayout.CENTER);
89      this.getContentPane().add(panel, BorderLayout.SOUTH);
90
91      scroll.setMaximumSize(new Dimension((w/2) + 300, (h/2) + 223));
92      scroll.setMinimumSize(new Dimension((w/2) + 300, (h/2) + 223));
93      scroll.setPreferredSize(new Dimension((w/2) + 300, (h/2) + 223));
94      this.setSize((w/2) + 300, (h/2) + 223);
95      this.setLocation((w - ((w/2) + 300))/2, (h - ((h/2) + 250))/2);
96      this.setResizable(false);
97      this.pack();
98      this.setVisible(true);
99      this.repaint();
100     this.validate();
101     }
102
103 public void actionPerformed(ActionEvent e)
104     {
105     if (e.getActionCommand().equals("back"))
106             {
107             this.setVisible(false);
108             }
109     }
110 }
```

```
1  //Title:        Sort
2  //Version:      1
3  //Copyright:    Copyright (c) 2001
4  //Author:       Audrey Lecerf
5  //Company:      MAI PROJECT - FUNDP
6  //Description:  Trier dans l'ordre alphabétique un vecteur de Strings
7
8
9  import java.awt.*;
10 import javax.swing.*;
11 import java.awt.event.*;
12 import java.util.*;
13
14 public class Sort
15 {
16 Vector sorted = new Vector();
17
18 public Sort(Vector v)
19     {
20     Vector vprime = (Vector)v.clone();
21
22     while (!vprime.isEmpty())
23         {
24         int x = 0;
25         String min = (String)vprime.get(x);
26         int indexmin = x;
27
28         int size = vprime.size();
29         while (x<size)
30             {
31             String temp = (String)vprime.get(x);
32             int less = temp.compareToIgnoreCase(min);
33             if (less < 0)
34                 {
35                 min = temp;
36                 indexmin = x;
37                 }
38             x++;
39             }
40         sorted.add(min);
41         vprime.removeElementAt(indexmin);
42         }
43     }
44
45 }
46
```

```
1  //Title:        Student
2  //Version:      1
3  //Copyright:    Copyright (c) 2001
4  //Author:       Audrey Lecerf
5  //Company:      MAI PROJECT - FUNDP
6  //Description:  Interface de l'élève
7
8
9  import java.awt.*;
10 import java.awt.event.*;
11
12 import javax.swing.*;
13 import javax.swing.event.*;
14 import javax.swing.border.*;
15
16 import java.lang.Object.*;
17
18 import java.util.*;
19
20 import java.rmi.*;
21
22 public class Student extends JPanel implements ActionListener
23     {
24     String title = "Etudiant";
25     String cours;
26     ServerRemote serv;
27     Term currentTerm;
28
29     /* Terms */
30
31     JTextField selectedterm = new JTextField();
32     JLabel terms = new JLabel("TERMES", JLabel.CENTER);
33     Vector vterms = new Vector();
34     Vector vuastemp = new Vector();
35     Vector vapptemp = new Vector();
36     Vector vanttemp = new Vector();
37     int maxlength = 0;
38     int maxuas = 0;
39     int maxapp = 0;
40     int maxant = 0;
41     boolean keypressed = false;
42
43     /* Definition Panel */
44
45     JPanel defPanel = new JPanel(new BorderLayout(10,10));
46     JPanel defnorth = new JPanel(new GridLayout(3,1));
47     JPanel defcenter = new JPanel(new BorderLayout());
48     JPanel abbpanel = new JPanel(new FlowLayout(FlowLayout.LEFT));
49     JPanel traducpanel = new JPanel(new FlowLayout(FlowLayout.LEFT));
50     JPanel defpanel = new JPanel(new FlowLayout(FlowLayout.LEFT));
51     JPanel boxes = new JPanel(new GridLayout(1,3,5,5));
52
53     JLabel term = new JLabel();
54     JLabel abb = new JLabel();
55     JLabel realabb = new JLabel("");
56     JLabel traduc = new JLabel();
57     JLabel twopoints = new JLabel();
58     JLabel realtraduc = new JLabel("");
59
60     JComboBox flag;
61     ImageIcon ndlflag;
62     ImageIcon englflag;
63     ImageIcon gerflag;
64     ImageIcon spanflag;
65
66     JLabel def = new JLabel();
67     JTextArea deftxt = new JTextArea(16, 45);
68
69     /* Boxes */
70
71     JLabel app = new JLabel("APPARENTES", JLabel.CENTER);
72     JLabel ant = new JLabel("OPPOSES", JLabel.CENTER);
73     JLabel uas = new JLabel("UA REFERENCEES", JLabel.CENTER);
74     Choice appbox;
75     Choice antbox;
76     Choice uasbox;
```

```
77      JButton seeua = new JButton("Aperçu");
78
79      GridBagLayout gbl = new GridBagLayout();
80      GridBagConstraints gblc = new GridBagConstraints();
81
82
83      public Student(ServerRemote s, String c)
84      {
85          try
86          {
87          serv = s;
88          cours = c;
89          displayPane();
90          }
91
92          catch(Exception ex)
93          {
94          ex.printStackTrace();
95          }
96      }
97
98      private void displayPane() throws RemoteException
99      {
100     vterms = (Vector)serv.loadTermNames(cours);
101     vuastemp = (Vector)serv.loaduaNV(cours);
102     vuastemp = (Vector)throwColumn(vuastemp);
103     vapptemp = (Vector)serv.loadAPP(cours);
104     vanttemp = (Vector)serv.loadANT(cours);
105     paintInterface();
106     }
107
108 /* Méthode qui dessine l'interface */
109
110 public void paintInterface()
111     {
112     /* Liste des Terms */
113
114         terms.setFont(new java.awt.Font("Dialog", 1, 12));
115         maxlength = maxlength(vterms);
116         maxuas = maxlength(vuastemp);
117         maxapp = maxlength(vapptemp);
118         maxant = maxlength(vanttemp);
119
120         final JList termlist = new JList(vterms);
121         termlist.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
122         JScrollPane termlistscroll = new JScrollPane(termlist);
123
            termlistscroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR
            _ALWAYS);
124
            termlistscroll.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROL
            LBAR_ALWAYS);
125
126         if (vterms.size() == 0)
127             {
128             termlistscroll.setMinimumSize(new Dimension(150, 530));
129             termlistscroll.setPreferredSize(new Dimension(150, 530));
130             termlistscroll.setMaximumSize(new Dimension(150, 530));
131             }
132         else
133             {
134             termlistscroll.setMinimumSize(new Dimension(max(150, maxlength*8,
                0), 530));
135             termlistscroll.setPreferredSize(new Dimension(max(150, maxlength*8,
                0), 530));
136             termlistscroll.setMaximumSize(new Dimension(max(150, maxlength*8,
                0), 530));
137             }
138
139         /* Lorsque l'utilisateur clique sur un mot, ce mot doit entrainer le  */
140         /* chargement du terme correspondant (accès BD), ce qui entrainera le */
141         /* remplissement des objets : newterm (nom du terme), appbox, antbox  */
142         /* et uasbos                                                          */
143
144     termlist.addListSelectionListener(new ListSelectionListener()
145     {
```

```
146     public void valueChanged(ListSelectionEvent e)
147         {
148         if (!keypressed)
149         {
150         int index = termlist.getSelectedIndex();
151
152                 if (index != -1)
153                     {
154                     if (currentTerm != null)
155                         {
156                         flag.setSelectedIndex(0);
157                         appbox.removeAll();
158                         antbox.removeAll();
159                         uasbox.removeAll();
160                         }
161
162                     String termdenom = (String)(vterms.elementAt(index));
163
164                     try
165                         {
166                             currentTerm = (Term)serv.loadTerm(termdenom, cours);
167                             term.setText(currentTerm.term);
168                             selectedterm.setText(currentTerm.term);
169                             realabb.setText(currentTerm.abbreviation);
170                             realtraduc.setText(currentTerm.englishtranslation);
171                             String d = currentTerm.definition.def;
172                             deftxt.setText(d);
173
174                             createboxes();
175
176                             setVisible(true);
177                             revalidate();
178                             repaint();
179
180                         }
181                     catch (RemoteException re)
182                         {
183                         re.printStackTrace();
184                         }
185                     }
186                 else if (vterms.size() == 0) new Msg("Le glossaire de ce cours
                    est vide");
187         }
188         keypressed = false;
189         }
190     });
191
192
193     termlist.addMouseListener(new MouseAdapter()
194     {
195
196         public void mouseClicked(MouseEvent e)
197     {
198         if (e.getClickCount() >= 1)
199             {
200             int index = termlist.locationToIndex(e.getPoint());
201
202             if (index != -1)
203                     {
204                     if (currentTerm != null)
205                         {
206                         flag.setSelectedIndex(0);
207                         appbox.removeAll();
208                         antbox.removeAll();
209                         uasbox.removeAll();
210                         }
211
212                     String termdenom = (String)(vterms.elementAt(index));
213
214                     try
215                         {
216                             currentTerm = (Term)serv.loadTerm(termdenom, cours);
217                             term.setText(currentTerm.term);
218                             selectedterm.setText(currentTerm.term);
219                             realabb.setText(currentTerm.abbreviation);
220                             realtraduc.setText(currentTerm.englishtranslation);
```

```
                        String d = currentTerm.definition.def;
                        deftxt.setText(d);

                        createboxes();

                        setVisible(true);
                        revalidate();
                        repaint();

                    }
                catch (RemoteException re)
                    {
                    re.printStackTrace();
                    }
                }
            else if (vterms.size() == 0) new Msg("Le glossaire de ce cours
            est vide");
        keypressed = false;
        }

    }
});


    selectedterm.addKeyListener(new UpdateList_KeyAdapter(this, termlist));

    /* Definition Panel */

    term.setFont(new java.awt.Font("Dialog", 1, 20));
    term.setBorder(BorderFactory.createEtchedBorder());
    term.setHorizontalAlignment(SwingConstants.CENTER);
    term.setBackground(Color.lightGray);
    term.setText("");
    defnorth.add(term);

    abb.setFont(new java.awt.Font("Dialog", 1, 12));
    abb.setText("ABREVIATION  :");
    abb.setBackground(Color.lightGray);
    abbpanel.add(abb);
    abbpanel.add(realabb);
    defnorth.add(abbpanel);

    /* traducpanel */

    traduc.setFont(new java.awt.Font("Dialog", 1, 12));
    traduc.setText("TRADUCTION ");
    traduc.setBackground(Color.lightGray);

    Flag fl = new Flag();
    flag = fl.f;

    flag.addActionListener(new ActionListener()
    {
        public void actionPerformed(ActionEvent e)
        {
        if (!((String)selectedterm.getText()).equals(""))
            {
            JComboBox cb = (JComboBox)e.getSource();
            ImageIcon newSelection = (ImageIcon)cb.getSelectedItem();

            String language = (String)newSelection.getDescription();

                if (language.startsWith("An"))
                    {
                    realtraduc.setText(currentTerm.englishtranslation);
                    }
                else if (language.startsWith("Né"))
                    {
                    realtraduc.setText(currentTerm.dutchtranslation);
                    }
                else if (language.startsWith("Al"))
                    {
                    realtraduc.setText(currentTerm.germantranslation);
                    }
                else if (language.startsWith("Es"))
```

```
                        {
                        realtraduc.setText(currentTerm.spanishtranslation);
                        }
                }
            }
            });

        twopoints.setFont(new java.awt.Font("Dialog", 1, 12));
        twopoints.setText(" : ");
        twopoints.setBackground(Color.lightGray);

        traducpanel.add(traduc);
        traducpanel.add(flag);
        traducpanel.add(twopoints);
        traducpanel.add(realtraduc);
        defnorth.add(traducpanel);

        deftxt.setBorder(new
        TitledBorder(LineBorder.createGrayLineBorder(),"DEFINITION"));
        deftxt.setEditable(false);
        deftxt.setLineWrap(true);
        deftxt.setWrapStyleWord(true);
        JScrollPane defscroll = new JScrollPane(deftxt);
        defcenter.add(defscroll, BorderLayout.CENTER);

        defPanel.add(defnorth, BorderLayout.NORTH);
        defPanel.add(defcenter, BorderLayout.CENTER);

        /* Boxes */

        app.setFont(new java.awt.Font("Dialog", 1, 12));
        ant.setFont(new java.awt.Font("Dialog", 1, 12));
        uas.setFont(new java.awt.Font("Dialog", 1, 12));

        appbox = new Choice();
        antbox = new Choice();
        uasbox = new Choice();

        boxes.setBackground(Color.orange);

        boxes.setMinimumSize(new Dimension(max((maxuas+13)*7*3, maxapp*7*3,
        maxant*7*3), 23));
        boxes.setPreferredSize(new Dimension(max((maxuas+13)*7*3, maxapp*7*3,
        maxant*7*3), 23));
        boxes.setMaximumSize(new Dimension(max((maxuas+13)*7*3, maxapp*7*3,
        maxant*7*3), 23));

        boxes.add(appbox);
        boxes.add(antbox);
        boxes.add(uasbox);

        seeua.setActionCommand("seeua");
        seeua.addActionListener(this);
        seeua.setToolTipText("Voir un aperçu de cette unité d'apprentissage");


        /* Main Panel : Arrangement des objets sur l'interface */

        setLayout(gbl);

        gblc.gridx = 0;
        gblc.gridy = 0;
        gblc.gridheight = 1;
        gblc.gridwidth = 1;
        gblc.insets = new Insets(5,5,5,5);
        gblc.fill = GridBagConstraints.HORIZONTAL;
        gblc.anchor = GridBagConstraints.CENTER;
        gbl.setConstraints(terms, gblc);
        add(terms);

        gblc.gridx = 1;
        gblc.gridy = 0;
        gblc.gridheight = 1;
        gblc.gridwidth = 1;
        gblc.weightx = 33;
        gbl.setConstraints(app, gblc);
```

```
368             add(app);
369
370             gblc.gridx = 2;
371             gblc.gridy = 0;
372             gblc.gridheight = 1;
373             gblc.gridwidth = 1;
374             gblc.weightx = 33;
375             gbl.setConstraints(ant, gblc);
376             add(ant);
377
378             gblc.gridx = 3;
379             gblc.gridy = 0;
380             gblc.gridheight = 1;
381             gblc.gridwidth = 1;
382             gblc.weightx = 33;
383             gbl.setConstraints(uas, gblc);
384             add(uas);
385
386             gblc.gridx = 0;
387             gblc.gridy = 1;
388             gblc.gridheight = 1;
389             gblc.gridwidth = 1;
390             gblc.weightx = 0;
391             gbl.setConstraints(selectedterm, gblc);
392             add(selectedterm);
393
394             gblc.gridx = 1;
395             gblc.gridy = 1;
396             gblc.gridwidth = 3;
397             gblc.gridheight = 1;
398             gbl.setConstraints(boxes, gblc);
399             add(boxes);
400
401             gblc.gridx = 4;
402             gblc.gridy = 1;
403             gblc.gridwidth = 1;
404             gblc.fill = GridBagConstraints.NONE;
405             gbl.setConstraints(seeua, gblc);
406             add(seeua);
407
408             gblc.gridx = 0;
409             gblc.gridy = 2;
410             gblc.fill = GridBagConstraints.BOTH;
411             gblc.gridheight = GridBagConstraints.REMAINDER;
412             gbl.setConstraints(termlistscroll, gblc);
413             add(termlistscroll);
414
415             gblc.gridx = 1;
416             gblc.gridy = 2;
417             gblc.gridwidth = GridBagConstraints.REMAINDER;
418             gblc.gridheight = GridBagConstraints.REMAINDER;
419             gbl.setConstraints(defPanel, gblc);
420             add(defPanel);
421
422             Toolkit t = Toolkit.getDefaultToolkit();
423             Dimension d = t.getScreenSize();
424             int h = d.height;
425             int w = d.width;
426             setMaximumSize(new Dimension(w/2 + 300, h/2 + 220));
427             setMinimumSize(new Dimension(w/2 + 300, h/2 + 220));
428             setPreferredSize(new Dimension(w/2 + 300, h/2 + 220));
429
430             setBackground(Color.orange);
431             setVisible(true);
432         }
433
434     /* Méthode qui crée les JComboBoxes des apparentés, opposés et uas */
435
436     public void createboxes()
437         {
438         Vector v_app = currentTerm.relatedterms;
439         Vector v_ant = currentTerm.antonyms;
440         Vector v_uas = currentTerm.uas;
441
442         if (v_app.size() != 0)
443                 {
```

```
                int c_app = 0;
                while (c_app < v_app.size())
                    {
                    String app_name = (String)v_app.get(c_app);
                    appbox.addItem(app_name);
                    c_app++;
                    }
                }


    if (v_ant.size() != 0)
                {
                int c_ant = 0;
                while (c_ant < v_ant.size())
                    {
                    String ant_name = (String)v_ant.get(c_ant);
                    antbox.addItem(ant_name);
                    c_ant++;
                    }
                }


    if (v_uas.size() != 0)
                {
                int c_uas = 0;
                while (c_uas < v_uas.size())
                    {
                    String uas_name = (String)((UA)(v_uas.get(c_uas))).titre + "
                    - version " +
                    (String)Integer.toString((int)((UA)(v_uas.get(c_uas))).versi
                    on);
                    uasbox.addItem(uas_name);
                    c_uas++;
                    }
                }

    }

/* Méthode qui ne garde que la première colonne d'un Vecteur */

Vector throwColumn(Vector v)
    {
    Vector vtc = new Vector();
    for (int x=0; x<v.size(); x++)
        {
        String temp = (String)(((Vector)v.get(x)).get(0));
        vtc.add(temp);
        }
    return vtc;
    }

/* Méthode qui indique si un Vector contient un certain String */

int contains(Vector v, String s)
    {
    int cont = -1;
    int size = v.size();
    int x = 0;
    boolean found = false;

    while ((x<size) && (!found) && (s.compareToIgnoreCase((String)v.get(x)) >=
    0))
        {
        if (((String)v.get(x)).equalsIgnoreCase(s))
            {
            cont = x;
            found = true;
            }
        x++;
        }
    if ((!found) && (x<size))
        {
        return x;
        }
    else return cont;
    }
```

```
516
517  /* Méthode qui renvoie la longueur maximale des éléments d'un vecteur */
518
519  int maxlength(Vector v)
520      {
521      int max = 0;
522      for (int x=0; x<v.size(); x++)
523          {
524          int temp = ((String)v.get(x)).length();
525          if (temp > max) max = temp;
526          }
527      return max;
528      }
529
530  /* Maximum entre 3 int */
531
532  int max(int a, int b, int c)
533      {
534      int tab[] = {a,b,c};
535      int max = 0;
536      for (int x=0 ; x<2 ; x++)
537          {
538          int temp = tab[x];
539          if (temp>max) max = temp;
540          }
541      return max;
542      }
543
544  /* Process the ENTER key pressing */
545
546   void selectedterm_keyPressed(java.awt.event.KeyEvent e, JList termlist)
547      {
548      if ((e.getKeyCode() == KeyEvent.VK_A) || (e.getKeyCode() == KeyEvent.VK_B)
       || (e.getKeyCode() == KeyEvent.VK_C)
549        || (e.getKeyCode() == KeyEvent.VK_D) || (e.getKeyCode() == KeyEvent.VK_E)
         || (e.getKeyCode() == KeyEvent.VK_F)
550        || (e.getKeyCode() == KeyEvent.VK_G) || (e.getKeyCode() == KeyEvent.VK_H)
         || (e.getKeyCode() == KeyEvent.VK_I)
551        || (e.getKeyCode() == KeyEvent.VK_J) || (e.getKeyCode() == KeyEvent.VK_K)
         || (e.getKeyCode() == KeyEvent.VK_L)
552        || (e.getKeyCode() == KeyEvent.VK_M) || (e.getKeyCode() == KeyEvent.VK_N)
         || (e.getKeyCode() == KeyEvent.VK_O)
553        || (e.getKeyCode() == KeyEvent.VK_P) || (e.getKeyCode() == KeyEvent.VK_Q)
         || (e.getKeyCode() == KeyEvent.VK_R)
554        || (e.getKeyCode() == KeyEvent.VK_S) || (e.getKeyCode() == KeyEvent.VK_T)
         || (e.getKeyCode() == KeyEvent.VK_U)
555        || (e.getKeyCode() == KeyEvent.VK_V) || (e.getKeyCode() == KeyEvent.VK_W)
         || (e.getKeyCode() == KeyEvent.VK_X)
556        || (e.getKeyCode() == KeyEvent.VK_Y) || (e.getKeyCode() == KeyEvent.VK_Z)
         || (e.getKeyCode() == KeyEvent.VK_CIRCUMFLEX)
557        || (e.getKeyCode() == KeyEvent.VK_MINUS) || (e.getKeyCode() ==
        KeyEvent.VK_UNDERSCORE) || (e.getKeyCode() == KeyEvent.VK_COMMA)
558        || (e.getKeyCode() == KeyEvent.VK_SPACE) || (e.getKeyCode() ==
        KeyEvent.VK_ROMAN_CHARACTERS))
559          {
560          String sterm = selectedterm.getText();
561          sterm = sterm + e.getKeyText(e.getKeyCode());
562          int ind = contains(vterms, sterm);
563          if (ind != -1)
564              {
565              keypressed = true;
566              termlist.setSelectedIndex(ind);
567              }
568          else
569              {
570              keypressed = true;
571              termlist.setSelectedIndex((vterms.size())-1);
572              }
573          }
574      }
575
576  /* Afficher l'UA sélectionnée */
577
578  public void actionPerformed(ActionEvent e)
579      {
580      String selectedua = (String)uasbox.getSelectedItem();
```

```
501      if (e.getActionCommand().equals("seeua") && selectedua != null)
502              {
503              int index = selectedua.indexOf(" - version ");
504          String s = (selectedua.substring(0, index)).trim();
505          String istring = (selectedua.substring(index+10)).trim();
506          int i = (int)Integer.parseInt(istring);
507
508          try
509              {
510              Vector vpages = (Vector)serv.loadPages(s, i);
511
512              if (vpages.size()>0)
513                  {
514                  ShowUA showua = new ShowUA(vpages);
515                  }
516              else
517                  {
518                  new Msg("Cette unité d'apprentissage est vide");
519                  }
520
521              /* Version Maurice */
522              /*
523              String cours = (String)serv.giveCours();
524              String domaine = (String)serv.giveDomaine();
525              Interface_Acceuil_1 accueil =
                 (Interface_Acceuil_1)serv.giveAccueil();
526              UA_Remote UA_Remote_Objects =
                 (UA_Remote)Naming.lookup("//138.48.32.193/servlets/UA_Implementation
                 ");
527              Unite_Apprentissage Mon_Objet_UA_Retourne =
                 UA_Remote_Objects.get_UA(s, cours, domaine);
528
529              JPanel Mon_Panel_Contenu_De_Navigation_UA =
                 accueil.Affiche_UA(Mon_Objet_UA_Retourne);
530
531              new ShowUA(Mon_Panel_Contenu_De_Navigation_UA, s);
532              */
533              }
534
535          catch (RemoteException ex)
536              {
537              ex.printStackTrace();
538              }
539
540          }
541      else if (uasbox.getItemCount() == 0) new Msg("Aucune UA n'est référencée par
         ce terme.");
542      else new Msg("Veuillez sélectionner une unité d'apprentissage.");
543      }
544  }
545
546
```

```
1  //Title:        Teacher
2  //Version:      1
3  //Copyright:    Copyright (c) 2001
4  //Author:       Audrey Lecerf
5  //Company:      MAI PROJECT - FUNDP
6  //Description:  Interface du professeur
7
8
9  import java.awt.*;
10 import java.awt.event.*;
11
12 import javax.swing.*;
13 import javax.swing.event.*;
14 import javax.swing.border.*;
15
16 import java.util.*;
17 import java.io.*;
18 import javax.swing.text.*;
19
20 import java.rmi.*;
21
22 public class Teacher extends JPanel implements ActionListener
23     {
24     final int maxuas = 4;
25
26     final String title = "Professeur";
27     String cours;
28     ServerRemote serv;
29     Vector addedterms = new Vector();
30
31     JPanel centerpanel = new JPanel();
32     JPanel southpanel = new JPanel();
33     JPanel panel = new JPanel(new GridLayout(2,1,5,5));
34
35     Vector vterms = new Vector();
36     JTextField newterm = new JTextField();
37     JLabel term = new JLabel();
38     JScrollPane termscroll;
39     JList termlist;
40
41     JLabel abb = new JLabel();
42     JTextField newabb = new JTextField();
43
44     JLabel traduc = new JLabel();
45     JComboBox flag;
46     String[] newtraducs = {"","","",""};
47     JTextField newtraduc = new JTextField();
48     JButton submit = new JButton("Soumettre");
49     JPanel traducpanel = new JPanel(new GridLayout(2,2,8,8));
50
51     JLabel app = new JLabel();
52     JTextField newapp = new JTextField();
53     Vector appvector = new Vector();
54     JList applist = new JList();
55     JScrollPane appscroll = new JScrollPane();
56     JButton addapp = new JButton("Ajouter");
57     JPanel apppanel = new JPanel(new GridLayout(1,1));
58     JButton toapp = new JButton();
59     JButton fromapp = new JButton();
60     JButton toallapp = new JButton();
61     JButton fromallapp = new JButton();
62
63     JLabel ant = new JLabel();
64     JTextField newant = new JTextField();
65     Vector antvector = new Vector();
66     JList antlist = new JList();
67     JScrollPane antscroll = new JScrollPane();
68     JButton addant = new JButton("Ajouter");
69     JPanel antpanel = new JPanel(new GridLayout(1,1));
70     JButton toant = new JButton();
71     JButton fromant = new JButton();
72     JButton toallant = new JButton();
73     JButton fromallant = new JButton();
74
75     JLabel def = new JLabel();
76     JPanel newdef = new JPanel(new BorderLayout());
```

```
77      JTextArea deftxt = new JTextArea();
78      JScrollPane scrolltxt;
79      JTextPane defimg = new JTextPane();
80      JScrollPane scrollimg;
81      JSplitPane split;
82      JButton insertimage = new JButton("Insérer une image");
83      JFileChooser jfch;
84      StyledEditorKit kit;
85      Document doc;
86      ImageIcon imageicon;
87      String description;
88
89      JLabel uasex = new JLabel();
90      Vector uasexvector = new Vector();
91      Vector uasextitle = new Vector();
92      noeditjtable uasexlist = new noeditjtable();
93      JScrollPane uasexscroll = new JScrollPane();
94      JPanel uasexpanel = new JPanel(new GridLayout(1,1));
95
96      JLabel uasref = new JLabel();
97      Vector uasrefvector = new Vector();
98      Vector uasreftitle = new Vector();
99      Vector uas = new Vector();
100     noeditjtable uasreflist = new noeditjtable();
101     JScrollPane uasrefscroll = new JScrollPane();
102     JPanel uasrefpanel = new JPanel(new GridLayout(1,1));
103
104     JButton touas = new JButton();
105     JButton fromuas = new JButton();
106     JButton toalluas = new JButton();
107     JButton fromalluas = new JButton();
108
109     JButton save = new JButton("Enregistrer le nouveau terme");
110     JButton delete = new JButton("Supprimer le terme sélectionné");
111     JButton deleteglossary = new JButton("Supprimer tout le glossaire");
112     JButton cancel = new JButton("Annuler");
113
114     GridBagLayout layout = new GridBagLayout();
115     GridBagConstraints constraints = new GridBagConstraints();
116
117     Definition defin;
118
119     public Teacher(ServerRemote s, String c)
120     {
121     try
122         {
123         serv = s;
124         cours = c;
125         displayPane();
126         }
127     catch(Exception ex)
128         {
129          ex.printStackTrace();
130         }
131     }
132
133     private void displayPane() throws RemoteException
134     {
135     vterms = (Vector)serv.loadTermNames(cours);
136     if (vterms.size() != 0) paintInterface();
137     else askWhatToDo();
138     }
139
140 /* Méthode qui demande que faire lorsque le glossaire est vide */
141
142 public void askWhatToDo()
143     {
144     Toolkit t = Toolkit.getDefaultToolkit();
145     Dimension d = t.getScreenSize();
146     int h = d.height;
147     int w = d.width;
148
149     JLabel question1 = new JLabel("Le glossaire de ce cours est vide.",
      JLabel.CENTER);
150     JLabel question2 = new JLabel("Désirez-vous créer un nouveau glossaire ou
      bien charger un glossaire existant ?", JLabel.CENTER);
```

```
151      JLabel vide1 = new JLabel("  ", JLabel.CENTER);
152      JLabel vide2 = new JLabel("  ", JLabel.CENTER);
153      JPanel q = new JPanel(new GridLayout(4,1));
154      JPanel labpanel = new JPanel(new BorderLayout());
155      JButton nouveau = new JButton("Créer un nouveau glossaire");
156      JButton existant = new JButton("Charger un glossaire existant");
157      JPanel boutons = new JPanel();
158
159      nouveau.setActionCommand("new");
160      nouveau.addActionListener(this);
161
162      existant.setActionCommand("ex");
163      existant.addActionListener(this);
164
165      question1.setFont(new java.awt.Font("Dialog", 1, 16));
166      question2.setFont(new java.awt.Font("Dialog", 1, 16));
167
168      q.setBackground(Color.orange);
169      labpanel.setBackground(Color.orange);
170      boutons.setBackground(Color.orange);
171      panel.setBackground(Color.orange);
172
173      panel.setMaximumSize(new Dimension(w/2, h/4));
174      panel.setMinimumSize(new Dimension(w/2, h/4));
175      panel.setPreferredSize(new Dimension(w/2, h/4));
176
177      q.add(question1);
178      q.add(vide1);
179      q.add(question2);
180      q.add(vide2);
181      boutons.add(nouveau);
182      boutons.add(existant);
183
184      labpanel.add(q, BorderLayout.SOUTH);
185      panel.add(labpanel);
186      panel.add(boutons);
187
188      setLayout(new BorderLayout());
189      add(panel, BorderLayout.CENTER);
190      setMaximumSize(new Dimension(w, h/2 + 220));
191      setMinimumSize(new Dimension(w/2 + 250, h/2 + 220));
192      setPreferredSize(new Dimension(w/2 + 250, h/2 + 220));
193      setBackground(Color.orange);
194      setVisible(true);
195      }
196
197  public void paintInterface()
198      {
199      /* Liste des Terms */
200
201      setLayout(new BorderLayout());
202      centerpanel.setLayout(layout);
203      centerpanel.setBackground(Color.orange);
204      southpanel.setLayout(new FlowLayout(FlowLayout.CENTER));
205      southpanel.setBackground(Color.orange);
206
207      /* Left panel : Liste des termes du glossaire, abbréviation,     */
208      /* traduction, apparentés et opposés                             */
209
210      term.setFont(new java.awt.Font("Dialog", 1, 12));
211      term.setHorizontalAlignment(SwingConstants.CENTER);
212      term.setText("                    TERMES                    ");
213
214      newterm.setText("Nouveau terme");
215      newterm.setHorizontalAlignment(SwingConstants.CENTER);
216      newterm.addMouseListener(new MouseAdapter()
217      {
218      public void mouseClicked(MouseEvent e)
219          {
220          if (e.getClickCount() >= 1 &&
          ((String)newterm.getText()).equals("Nouveau terme"))
221                  {
222                  newterm.setText("");
223                  }
224          }
225      });
```

```
226
227         int maxlength = maxlength(vterms);
228
229         termlist = new JList(vterms);
230         termlist.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
231         termscroll = new JScrollPane(termlist);
232
            termscroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS)
            ;
233
            termscroll.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_ALW
            AYS);
234         termscroll.setMinimumSize(new Dimension(maxlength,800));
235         termscroll.setPreferredSize(new Dimension(maxlength,800));
236         termscroll.setMaximumSize(new Dimension(maxlength,800));
237         termlist.setSelectedIndex(0);
238
239         newterm.addKeyListener(new UpdateList_KeyAdapter(this, termlist, vterms,
            newterm));
240
241         abb.setFont(new java.awt.Font("Dialog", 1, 12));
242         abb.setHorizontalAlignment(SwingConstants.CENTER);
243         abb.setText("ABREVIATION");
244
245         newabb.setText("Abréviation du nouveau terme");
246         newabb.setHorizontalAlignment(SwingConstants.CENTER);
247         newabb.addMouseListener(new MouseAdapter()
248         {
249         public void mouseClicked(MouseEvent e)
250             {
251             if (e.getClickCount() >= 1 &&
            ((String)newabb.getText()).equals("Abréviation du nouveau terme"))
252                     {
253                     newabb.setText("");
254                     }
255             }
256         });
257
258         traduc.setFont(new java.awt.Font("Dialog", 1, 12));
259         traduc.setHorizontalAlignment(SwingConstants.CENTER);
260         traduc.setText("      TRADUCTION :      ");
261
262         Flag fl = new Flag();
263         flag = fl.f;
264
265         flag.addActionListener(new ActionListener()
266         {
267             public void actionPerformed(ActionEvent e)
268             {
269             JComboBox cb = (JComboBox)e.getSource();
270             ImageIcon newSelection = (ImageIcon)cb.getSelectedItem();

271             String language = (String)newSelection.getDescription();
272
273             if (language.startsWith("An"))
274                 {
275                 newtraduc.setText(newtraducs[0]);
276                 }
277             else if (language.startsWith("Né"))
278                 {
279                 newtraduc.setText(newtraducs[1]);
280                 }
281             else if (language.startsWith("Al"))
282                 {
283                 newtraduc.setText(newtraducs[2]);
284                 }
285             else if (language.startsWith("Es"))
286                 {
287                 newtraduc.setText(newtraducs[3]);
288                 }
289             }
290             });
291
292         newtraduc.setText("Nouvelle traduction");
293         newtraduc.setHorizontalAlignment(SwingConstants.CENTER);
294         newtraduc.addMouseListener(new MouseAdapter()
230
```

```
295     {
296     public void mouseClicked(MouseEvent e)
297         {
298         if (e.getClickCount() >= 1 &&
        ((String)newtraduc.getText()).equals("Nouvelle traduction"))
299                 {
300                 newtraduc.setText("");
301                 }
302         }
303     });
304
305     submit.setActionCommand("submit");
306     submit.addActionListener(this);
307
308     traducpanel.setBackground(Color.orange);
309     traducpanel.add(traduc);
310     traducpanel.add(newtraduc);
311     traducpanel.add(flag);
312     traducpanel.add(submit);
313
314     app.setHorizontalAlignment(SwingConstants.CENTER);
315     app.setFont(new java.awt.Font("Dialog", 1, 12));
316     app.setText("APPARENTES");
317     newapp.setHorizontalAlignment(SwingConstants.CENTER);
318     newapp.setText("Nouvel apparenté");
319     newapp.addMouseListener(new MouseAdapter()
320     {
321     public void mouseClicked(MouseEvent e)
322         {
323         if (e.getClickCount() >= 1 && ((String)newapp.getText()).equals("Nouvel
        apparenté"))
324                 {
325                 newapp.setText("");
326                 }
327         }
328     });
329
330     addapp.setActionCommand("addapp");
331     addapp.addActionListener(this);
332
333     apppanel.add(appscroll);
334
335     toapp.setFont(new java.awt.Font("Dialog", 1, 15));
336     fromapp.setFont(new java.awt.Font("Dialog", 1, 15));
337     toallapp.setFont(new java.awt.Font("Dialog", 1, 15));
338     fromallapp.setFont(new java.awt.Font("Dialog", 1, 15));
339
340     toapp.setText(">");
341     fromapp.setText("<");
342     toallapp.setText(">>");
343     fromallapp.setText("<<");
344
345     toapp.setMargin(new Insets(2,7,2,7));
346     fromapp.setMargin(new Insets(2,7,2,7));
347     toallapp.setMargin(new Insets(2,2,2,2));
348     fromallapp.setMargin(new Insets(2,2,2,2));
349
350     toapp.setToolTipText("Insérer ce terme dans la liste des termes
        apparentés");
351     fromapp.setToolTipText("Supprimer ce terme de la liste des termes
        apparentés");
352     toallapp.setToolTipText("Insérer tous les termes du glossaire dans la liste
        des termes apparentés");
353     fromallapp.setToolTipText("Supprimer tous les termes de la liste des termes
        apparentés");
354
355     toapp.setActionCommand("toapp");
356     toapp.addActionListener(this);
357
358     fromapp.setActionCommand("fromapp");
359     fromapp.addActionListener(this);
360
361     toallapp.setActionCommand("toallapp");
362     toallapp.addActionListener(this);
363
364     fromallapp.setActionCommand("fromallapp");
```

```
365       fromallapp.addActionListener(this);
366
367       ant.setHorizontalAlignment(SwingConstants.CENTER);
368       ant.setFont(new java.awt.Font("Dialog", 1, 12));
369       ant.setText("OPPOSES");
370
371       newant.setHorizontalAlignment(SwingConstants.CENTER);
372       newant.setText("Nouvel opposé");
373       newant.addMouseListener(new MouseAdapter()
374       {
375       public void mouseClicked(MouseEvent e)
376           {
377           if (e.getClickCount() >= 1 && ((String)newant.getText()).equals("Nouvel
          opposé"))
378                   {
379                   newant.setText("");
380                   }
381           }
382       });
383
384       addant.setActionCommand("addant");
385       addant.addActionListener(this);
386
387       antpanel.add(antscroll);
388
389       toant.setFont(new java.awt.Font("Dialog", 1, 15));
390       fromant.setFont(new java.awt.Font("Dialog", 1, 15));
391       toallant.setFont(new java.awt.Font("Dialog", 1, 15));
392       fromallant.setFont(new java.awt.Font("Dialog", 1, 15));
393       toant.setText(">");
394       fromant.setText("<");
395       toallant.setText(">>");
396       fromallant.setText("<<");
397       toant.setMargin(new Insets(2,7,2,7));
398       fromant.setMargin(new Insets(2,7,2,7));
399       toallant.setMargin(new Insets(2,2,2,2));
400       fromallant.setMargin(new Insets(2,2,2,2));
401
402       toant.setToolTipText("Insérer ce terme dans la liste des termes opposés");
403       fromant.setToolTipText("Supprimer ce terme de la liste des termes opposés");
404       toallant.setToolTipText("Insérer tous les termes du glossaire dans la liste
          des termes opposés");
405       fromallant.setToolTipText("Supprimer tous les termes de la liste des termes
          opposés");
406
407       toant.setActionCommand("toant");
408       toant.addActionListener(this);
409
410       fromant.setActionCommand("fromant");
411       fromant.addActionListener(this);
412
413       toallant.setActionCommand("toallant");
414       toallant.addActionListener(this);
415
416       fromallant.setActionCommand("fromallant");
417       fromallant.addActionListener(this);
418
419       /* Arrangement des objets sur la partie gauche de l'interface */
420
421       constraints.gridx = 0;
422       constraints.gridy = 0;
423       constraints.gridheight = 1;
424       constraints.gridwidth = 1;
425       constraints.insets = new Insets(4,4,4,4);
426       constraints.fill = GridBagConstraints.HORIZONTAL;
427       layout.setConstraints(term, constraints);
428       centerpanel.add(term);
429
430       constraints.gridx = 0;
431       constraints.gridy = 1;
432       constraints.gridheight = 1;
433       constraints.gridwidth = 1;
434       layout.setConstraints(newterm, constraints);
435       centerpanel.add(newterm);
436
437       constraints.gridx = 0;
```

```
438         constraints.gridy = 2;
439         constraints.gridheight = 14;
440         constraints.gridwidth = 1;
441         constraints.weighty = 100;
442         constraints.fill = GridBagConstraints.BOTH;
443         layout.setConstraints(termscroll, constraints);
444         centerpanel.add(termscroll);
445
446         constraints.gridx = 1;
447         constraints.gridy = 0;
448         constraints.gridheight = 1;
449         constraints.gridwidth = 3;
450         constraints.weighty = 0;
451         constraints.fill = GridBagConstraints.HORIZONTAL;
452         layout.setConstraints(abb, constraints);
453         centerpanel.add(abb);
454
455         constraints.gridx = 1;
456         constraints.gridy = 1;
457         constraints.gridheight = 1;
458         constraints.gridwidth = 3;
459         constraints.fill = GridBagConstraints.BOTH;
460         layout.setConstraints(newabb, constraints);
461         centerpanel.add(newabb);
462
463         constraints.gridx = 1;
464         constraints.gridy = 2;
465         constraints.gridheight = 2;
466         constraints.gridwidth = 3;
467         constraints.fill = GridBagConstraints.HORIZONTAL;
468         layout.setConstraints(traducpanel, constraints);
469         centerpanel.add(traducpanel);
470
471         constraints.gridx = 2;
472         constraints.gridy = 4;
473         constraints.gridheight = 1;
474         constraints.gridwidth = 2;
475         constraints.fill = GridBagConstraints.HORIZONTAL;
476         layout.setConstraints(app, constraints);
477         centerpanel.add(app);
478
479         constraints.gridx = 2;
480         constraints.gridy = 5;
481         constraints.gridheight = 1;
482         constraints.gridwidth = 1;
483         constraints.ipadx = 125;
484         constraints.fill = GridBagConstraints.BOTH;
485         layout.setConstraints(newapp, constraints);
486         centerpanel.add(newapp);
487
488         constraints.gridx = 3;
489         constraints.gridy = 5;
490         constraints.gridheight = 1;
491         constraints.gridwidth = 1;
492         constraints.ipadx = 0;
493         constraints.anchor = GridBagConstraints.EAST;
494         constraints.fill = GridBagConstraints.NONE;
495         layout.setConstraints(addapp, constraints);
496         centerpanel.add(addapp);
497
498         constraints.gridx = 1;
499         constraints.gridy = 6;
500         constraints.gridheight = 1;
501         constraints.gridwidth = 1;
502         constraints.anchor = GridBagConstraints.CENTER;
503         constraints.fill = GridBagConstraints.NONE;
504         layout.setConstraints(toapp, constraints);
505         centerpanel.add(toapp);
506
507         constraints.gridx = 1;
508         constraints.gridy = 7;
509         constraints.gridheight = 1;
510         constraints.gridwidth = 1;
511         layout.setConstraints(fromapp, constraints);
512         centerpanel.add(fromapp);
513
```

```
514         constraints.gridx = 1;
515         constraints.gridy = 8;
516         constraints.gridheight = 1;
517         constraints.gridwidth = 1;
518         layout.setConstraints(toallapp, constraints);
519         centerpanel.add(toallapp);
520
521         constraints.gridx = 1;
522         constraints.gridy = 9;
523         constraints.gridheight = 1;
524         constraints.gridwidth = 1;
525         layout.setConstraints(fromallapp, constraints);
526         centerpanel.add(fromallapp);
527
528         constraints.gridx = 2;
529         constraints.gridy = 6;
530         constraints.gridheight = 4;
531         constraints.gridwidth = 2;
532         constraints.fill = GridBagConstraints.BOTH;
533         layout.setConstraints(apppanel, constraints);
534         centerpanel.add(apppanel);
535
536         constraints.gridx = 2;
537         constraints.gridy = 10;
538         constraints.gridheight = 1;
539         constraints.gridwidth = 2;
540         constraints.fill = GridBagConstraints.HORIZONTAL;
541         layout.setConstraints(ant, constraints);
542         centerpanel.add(ant);
543
544         constraints.gridx = 2;
545         constraints.gridy = 11;
546         constraints.gridheight = 1;
547         constraints.gridwidth = 1;
548         constraints.fill = GridBagConstraints.BOTH;
549         layout.setConstraints(newant, constraints);
550         centerpanel.add(newant);
551
552         constraints.gridx = 3;
553         constraints.gridy = 11;
554         constraints.gridheight = 1;
555         constraints.gridwidth = 1;
556         constraints.anchor = GridBagConstraints.EAST;
557         constraints.fill = GridBagConstraints.NONE;
558         layout.setConstraints(addant, constraints);
559         centerpanel.add(addant);
560
561         constraints.gridx = 1;
562         constraints.gridy = 12;
563         constraints.gridheight = 1;
564         constraints.gridwidth = 1;
565         constraints.anchor = GridBagConstraints.CENTER;
566         constraints.fill = GridBagConstraints.NONE;
567         layout.setConstraints(toant, constraints);
568         centerpanel.add(toant);
569
570         constraints.gridx = 1;
571         constraints.gridy = 13;
572         constraints.gridheight = 1;
573         constraints.gridwidth = 1;
574         layout.setConstraints(fromant, constraints);
575         centerpanel.add(fromant);
576
577         constraints.gridx = 1;
578         constraints.gridy = 14;
579         constraints.gridheight = 1;
580         constraints.gridwidth = 1;
581         layout.setConstraints(toallant, constraints);
582         centerpanel.add(toallant);
583
584         constraints.gridx = 1;
585         constraints.gridy = 15;
586         constraints.gridheight = 1;
587         constraints.gridwidth = 1;
588         layout.setConstraints(fromallant, constraints);
589         centerpanel.add(fromallant);
```

```
590
591         constraints.gridx = 2;
592         constraints.gridy = 12;
593         constraints.gridheight = 4;
594         constraints.gridwidth = 2;
595         constraints.fill = GridBagConstraints.BOTH;
596         layout.setConstraints(antpanel, constraints);
597         centerpanel.add(antpanel);
598
599         /* Right panel : nouvelle définition et uas référencées */
600
601         def.setText("DEFINITION");
602         def.setFont(new java.awt.Font("Dialog", 1, 12));
603         def.setHorizontalAlignment(SwingConstants.CENTER);
604
605         deftxt.setEditable(true);
606         deftxt.setLineWrap(true);
607         deftxt.setWrapStyleWord(true);
608         defimg.setEditable(false);
609
610         scrolltxt = new JScrollPane(deftxt);
611         scrollimg = new JScrollPane(defimg);
612         scrolltxt.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
613
            scrolltxt.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_ALWA
            YS);
614         scrollimg.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
615
            scrollimg.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_ALWA
            YS);
616
617         split = new JSplitPane(JSplitPane.VERTICAL_SPLIT, scrolltxt, scrollimg);
618         split.setDividerLocation(150);
619         split.setBackground(Color.orange);
620         split.setForeground(Color.orange);
621         newdef.add(split, BorderLayout.CENTER);
622
623         insertimage.setActionCommand("insertimage");
624         insertimage.addActionListener(this);
625
626         jfch = new JFileChooser();
627         jfch.setCurrentDirectory(new File("."));
628         SimpleFilter filtergif = new SimpleFilter("gif", "GIF images");
629         SimpleFilter filterjpg = new SimpleFilter("jpg", "JPG images");
630         SimpleFilter filterjpeg = new SimpleFilter("jpeg", "JPEG images");
631         jfch.addChoosableFileFilter(filtergif);
632         jfch.addChoosableFileFilter(filterjpg);
633         jfch.addChoosableFileFilter(filterjpeg);
634         jfch.setFileFilter(filtergif);
635         kit = new StyledEditorKit();
636         doc = (Document)(kit.createDefaultDocument());
637         defimg.setEditorKit(kit);
638         defimg.setDocument(doc);
639
640         uasex.setText("UA EXISTANTES");
641         uasex.setFont(new java.awt.Font("Dialog", 1, 12));
642         uasex.setHorizontalAlignment(SwingConstants.CENTER);
643
644         uasextitle.add("Titre");
645         uasextitle.add("Version");
646         try
647             {
648             uasexvector = (Vector)serv.loaduaNV(cours);
649             }
650         catch (RemoteException re)
651             {
652             re.printStackTrace();
653             }
654         uasexlist = new noeditjtable(uasexvector, uasextitle);
655         uasexlist.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
656         uasexscroll = new JScrollPane(uasexlist);
657
658         uasexpanel.add(uasexscroll);
659
660         uasref.setText("UA REFERENCEES");
661         uasref.setFont(new java.awt.Font("Dialog", 1, 12));
```

```
662       uasref.setHorizontalAlignment(SwingConstants.CENTER);
663
664       uasreftitle.add("Titre");
665       uasreftitle.add("Version");
666       uasreflist = new noeditjtable(uasrefvector, uasreftitle);
667       uasreflist.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
668       uasrefscroll = new JScrollPane(uasreflist);
669
670       uasrefpanel.add(uasrefscroll);
671
672       touas.setText(">");
673       touas.setFont(new java.awt.Font("Dialog", 1, 15));
674       fromuas.setFont(new java.awt.Font("Dialog", 1, 15));
675       fromuas.setText("<");
676       toalluas.setText(">>");
677       toalluas.setFont(new java.awt.Font("Dialog", 1, 15));
678       fromalluas.setFont(new java.awt.Font("Dialog", 1, 15));
679       fromalluas.setText("<<");
680
681       touas.setMargin(new Insets(2,7,2,7));
682       fromuas.setMargin(new Insets(2,7,2,7));
683       toalluas.setMargin(new Insets(2,2,2,2));
684       fromalluas.setMargin(new Insets(2,2,2,2));
685
686       touas.setToolTipText("Insérer cette unité d'apprentissage dans la liste des
          unités d'apprentissage");
687       fromuas.setToolTipText("Supprimer cette unité d'apprentissage de la liste
          des unités d'apprentissage");
688       toalluas.setToolTipText("Insérer toutes les unités d'apprentissage dans la
          liste des unités d'apprentissage");
689       fromalluas.setToolTipText("Supprimer toutes les unités d'apprentissage de la
          liste des unités d'apprentissage");
690
691       touas.setActionCommand("touas");
692       touas.addActionListener(this);
693
694       fromuas.setActionCommand("fromuas");
695       fromuas.addActionListener(this);
696
697       toalluas.setActionCommand("toalluas");
698       toalluas.addActionListener(this);
699
700       fromalluas.setActionCommand("fromalluas");
701       fromalluas.addActionListener(this);
702
703       save.setActionCommand("save");
704       save.addActionListener(this);
705
706       delete.setActionCommand("delete");
707       delete.addActionListener(this);
708
709       deleteglossary.setActionCommand("deleteglossary");
710       deleteglossary.addActionListener(this);
711
712       cancel.setActionCommand("cancel");
713       cancel.addActionListener(this);
714
715       /* Arrangement des objets sur la partie droite de l'interface */
716
717       constraints.gridx = 4;
718       constraints.gridy = 0;
719       constraints.gridheight = 1;
720       constraints.gridwidth = 5;
721       constraints.fill = GridBagConstraints.HORIZONTAL;
722       layout.setConstraints(def, constraints);
723       centerpanel.add(def);
724
725       constraints.gridx = 4;
726       constraints.gridy = 1;
727       constraints.gridheight = 9;
728       constraints.gridwidth = 5;
729       constraints.fill = GridBagConstraints.BOTH;
730       layout.setConstraints(newdef, constraints);
731       centerpanel.add(newdef);
732
733       constraints.gridx = 4;
```

```
734         constraints.gridy = 10;
735         constraints.gridheight = 1;
736         constraints.gridwidth = 5;
737         constraints.fill = GridBagConstraints.BOTH;
738         layout.setConstraints(insertimage, constraints);
739         centerpanel.add(insertimage);
740
741         constraints.gridx = 4;
742         constraints.gridy = 11;
743         constraints.gridheight = 1;
744         constraints.gridwidth = 2;
745         constraints.fill = GridBagConstraints.HORIZONTAL;
746         layout.setConstraints(uasex, constraints);
747         centerpanel.add(uasex);
748
749         constraints.gridx = 4;
750         constraints.gridy = 12;
751         constraints.gridheight = 4;
752         constraints.gridwidth = 2;
753         constraints.weightx = 50;
754         constraints.fill = GridBagConstraints.BOTH;
755         layout.setConstraints(uasexpanel, constraints);
756         centerpanel.add(uasexpanel);
757
758         constraints.gridx = 6;
759         constraints.gridy = 12;
760         constraints.gridheight = 1;
761         constraints.gridwidth = 1;
762         constraints.weightx = 0;
763         constraints.fill = GridBagConstraints.NONE;
764         layout.setConstraints(touas, constraints);
765         centerpanel.add(touas);
766
767         constraints.gridx = 6;
768         constraints.gridy = 13;
769         constraints.gridheight = 1;
770         constraints.gridwidth = 1;
771         layout.setConstraints(fromuas, constraints);
772         centerpanel.add(fromuas);
773
774         constraints.gridx = 6;
775         constraints.gridy = 14;
776         constraints.gridheight = 1;
777         constraints.gridwidth = 1;
778         layout.setConstraints(toalluas, constraints);
779         centerpanel.add(toalluas);
780
781         constraints.gridx = 6;
782         constraints.gridy = 15;
783         constraints.gridheight = 1;
784         constraints.gridwidth = 1;
785         layout.setConstraints(fromalluas, constraints);
786         centerpanel.add(fromalluas);
787
788         constraints.gridx = 7;
789         constraints.gridy = 11;
790         constraints.gridheight = 1;
791         constraints.gridwidth = 2;
792         constraints.fill = GridBagConstraints.HORIZONTAL;
793         layout.setConstraints(uasref, constraints);
794         centerpanel.add(uasref);
795
796         constraints.gridx = 7;
797         constraints.gridy = 12;
798         constraints.gridheight = 4;
799         constraints.gridwidth = 2;
800         constraints.weightx = 50;
801         constraints.fill = GridBagConstraints.BOTH;
802         layout.setConstraints(uasrefpanel, constraints);
803         centerpanel.add(uasrefpanel);
804
805
806         southpanel.add(save);
807         southpanel.add(delete);
808         southpanel.add(deleteglossary);
809
```

```
910       add(centerpanel, BorderLayout.CENTER);
911       add(southpanel, BorderLayout.SOUTH);
912
913       /* Main panel */
914
915
916       Toolkit t = Toolkit.getDefaultToolkit();
917       Dimension d = t.getScreenSize();
918       int h = d.height;
919       int w = d.width;
920       setMaximumSize(new Dimension(w, h/2 + 220));
921       setMinimumSize(new Dimension(w/2 + 250, h/2 + 220));
922       setPreferredSize(new Dimension(w/2 + 250, h/2 + 220));
923
924       setBackground(Color.orange);
925       setVisible(true);
926       validate();
927       repaint();
928       }
929
930   /* La méthode index donne l'indice de l'endroit où se trouve un String dans un
      Vector */
931
932    int index(Vector v, String s)
933       {
934       int index = -1;
935       int size = v.size();
936       int x = 0;
937       boolean found = false;
938
939       while ((x<size) && (!found))
940           {
941           if (((String)v.get(x)).equalsIgnoreCase(s))
942               {
943               index = x;
944               found = true;
945               }
946           x++;
947           }
948       return index;
949       }
950
951   /* La méthode contains donne la position d'une certain String dans un Vector */
952
953    int contains(Vector v, String s)
954       {
955       int cont = -1;
956       int size = v.size();
957       int x = 0;
958       boolean found = false;
959
960       while ((x<size) && (!found) && (s.compareToIgnoreCase((String)v.get(x)) >=
          0))
961           {
962           if (((String)v.get(x)).equalsIgnoreCase(s))
963               {
964               cont = x;
965               found = true;
966               }
967           x++;
968           }
969       if ((!found) && (x<size))
970           {
971           return x;
972           }
973       else return cont;
974       }
975
976   /* La méthode contains indique si un Vector contient un certain String */
977
978    boolean containsYorN(Vector v, String s)
979       {
980       int size = v.size();
981       int x = 0;
982       boolean found = false;
983
```

```
984         while ((x<size) && (!found) && (s.compareToIgnoreCase((String)v.get(x)) >=
            0))
985             {
986             if (((String)v.get(x)).equalsIgnoreCase(s))
987                 {
988                 found = true;
989                 }
990             x++;
991             }
992         return found;
993         }
994
995         /* Mettre à jour la liste des termes existants dans le glossaire */
996
997     public void update() throws RemoteException
998         {
999         vterms.clear();
1000        vterms = (Vector)serv.loadTermNames(cours);
1001        termlist = new JList(vterms);
1002        termlist.repaint();
1003        termlist.revalidate();
1004        }
1005
1006    /* Change a string containing a "'" into another string that can be accepted by
        a SQL query */
1007
1008    public String changeAp(String are)
1009        {
1010        String neware = are;
1011        int indof = neware.indexOf("'", 0);
1012
1013        while (indof >=0)
1014            {
1015            String start = neware.substring(0, indof);
1016            String end = neware.substring(indof+1);
1017            neware = start + "''" + end;
1018            indof = neware.indexOf("'", indof+2);
1019            }
1020
1021        return neware;
1022        }
1023
1024    /* Méthode qui renvoie la longueur maximale des éléments d'un vecteur */
1025
1026    int maxlength(Vector v)
1027        {
1028        int max = 0;
1029        for (int x=0; x<v.size(); x++)
1030            {
1031            int temp = ((String)v.get(x)).length();
1032            if (temp > max) max = temp;
1033            }
1034        return max;
1035        }
1036
1037     /* Process the ENTER key pressing */
1038
1039     void new_keyPressed(java.awt.event.KeyEvent e, JList list, Vector v, JTextField
         jtf)
1040        {
1041        if ((e.getKeyCode() == KeyEvent.VK_A) || (e.getKeyCode() == KeyEvent.VK_B)
           || (e.getKeyCode() == KeyEvent.VK_C)
1042           || (e.getKeyCode() == KeyEvent.VK_D) || (e.getKeyCode() == KeyEvent.VK_E)
           || (e.getKeyCode() == KeyEvent.VK_F)
1043           || (e.getKeyCode() == KeyEvent.VK_G) || (e.getKeyCode() == KeyEvent.VK_H)
           || (e.getKeyCode() == KeyEvent.VK_I)
1044           || (e.getKeyCode() == KeyEvent.VK_J) || (e.getKeyCode() == KeyEvent.VK_K)
           || (e.getKeyCode() == KeyEvent.VK_L)
1045           || (e.getKeyCode() == KeyEvent.VK_M) || (e.getKeyCode() == KeyEvent.VK_N)
           || (e.getKeyCode() == KeyEvent.VK_O)
1046           || (e.getKeyCode() == KeyEvent.VK_P) || (e.getKeyCode() == KeyEvent.VK_Q)
           || (e.getKeyCode() == KeyEvent.VK_R)
1047           || (e.getKeyCode() == KeyEvent.VK_S) || (e.getKeyCode() == KeyEvent.VK_T)
           || (e.getKeyCode() == KeyEvent.VK_U)
1048           || (e.getKeyCode() == KeyEvent.VK_V) || (e.getKeyCode() == KeyEvent.VK_W)
           || (e.getKeyCode() == KeyEvent.VK_X)
```

```
949         || (e.getKeyCode() == KeyEvent.VK_Y) || (e.getKeyCode() == KeyEvent.VK_Z))
950             {
951             String s = jtf.getText();
952             s = s + e.getKeyText(e.getKeyCode());
953             int ind = contains(v, s);
954             if (ind != -1) list.setSelectedIndex(ind);
955             else if (!v.isEmpty())
956                 {
957                 list.setSelectedIndex((v.size())-1);
958                 }
959             }
960         }
961
962
963         public void actionPerformed(ActionEvent e)
964         {
965         /* Ajout d'un apparenté */
966
967         if (e.getActionCommand().equals("addapp"))
968             {
969             String napp = newapp.getText();
970             napp = napp.trim();
971
972             if (napp.equals("Nouvel apparenté") || napp.equals(""))
973                 {
974                 /* Pas de NOUVEAU terme dans le textfield */
975                 new Msg("Le terme apparenté est invalide.");
976                 }
977             else
978                 {
979                 /* Cas où la liste est vide */
980                 if (appvector.isEmpty())
981                         {
982                         apppanel.remove(appscroll);
983                         appvector.add(napp);
984                         applist = new JList(appvector);
985                         appscroll = new JScrollPane(applist);
986                         apppanel.add(appscroll);
987                         }
988
989                 /* Cas où la liste n'est pas vide */
990                 else
991                         {
992                         int i = index(appvector, napp);
993
994                         /* Cas où le terme ne se trouve pas encore dans la liste
                            (vecteur)*/
995                         if (i == -1)
996                             {
997                             /* On enlève la liste pour la trier (après y avoir
                                ajouté le   */
998                             /* nouveau terme). Ensuite, on le replace dans le panel.
                                    */
999                             apppanel.remove(appscroll);
1000                            appvector.add(napp);
1001                            Sort sort = new Sort(appvector);
1002                            appvector = sort.sorted;
1003                            applist = new JList(appvector);
1004                            appscroll = new JScrollPane(applist);
1005                            apppanel.add(appscroll);
1006                            }
1007
1008                        /* Cas où le terme se trouve déjà dans la liste (vecteur)*/
1009                        else new Msg("Ce terme a déjà été ajouté.");
1010                        }
1011                }
1012            newapp.setText("Nouvel apparenté");
1013            apppanel.validate();
1014            apppanel.repaint();
1015            }
1016
1017        /* Ajout d'un opposé */
1018
1019        else if (e.getActionCommand().equals("addant"))
1020            {
1021            String nant = newant.getText();
```

```
1022            nant = nant.trim();
1023
1024            if (nant.equals("Nouvel opposé") || nant.equals(""))
1025                {
1026                /* Pas de NOUVEAU terme dans le textfield */
1027                new Msg("L'opposé est invalide.");
1028                }
1029            else
1030                {
1031                /* Cas où la liste est vide */
1032                if (antvector.isEmpty())
1033                        {
1034                        antpanel.remove(antscroll);
1035                        antvector.add(nant);
1036                        antlist = new JList(antvector);
1037                        antscroll = new JScrollPane(antlist);
1038                        antpanel.add(antscroll);
1039                        }
1040
1041                /* Cas où la liste n'est pas vide */
1042                else
1043                        {
1044                        int i = index(antvector, nant);
1045
1046                        /* Cas où le terme ne se trouve pas encore dans la liste
                        (vecteur)  */
1047                        if (i == -1)
1048                            {
1049                            /* On enlève la liste pour la trier (après y avoir   */
1050                            /* ajouté le nouveau terme). Ensuite, on le replace */
1051                            /* dans le panel.                                    */
1052                            antpanel.remove(antscroll);
1053                            antvector.add(nant);
1054                            Sort sort = new Sort(antvector);
1055                            antvector = sort.sorted;
1056                            antlist = new JList(antvector);
1057                            antscroll = new JScrollPane(antlist);
1058                            antpanel.add(antscroll);
1059                            }
1060
1061                        /* Cas où le terme se trouve déjà dans la liste (vecteur)*/
1062                        else new Msg("Ce terme a déjà été ajouté.");
1063                        }
1064                }
1065            newant.setText("Nouvel opposé");
1066            antpanel.validate();
1067            antpanel.repaint();
1068            }
1069
1070        /* Ajouter un terme de la liste des termes dans la liste des apparentés */
1071
1072        else if (e.getActionCommand().equals("toapp"))
1073            {
1074            String sterm = (String)termlist.getSelectedValue();
1075
1076            if (sterm != null)
1077                {
1078                if (appvector.isEmpty())
1079                    {
1080                    apppanel.remove(appscroll);
1081                    appvector.add(sterm);
1082                    applist = new JList(appvector);
1083                    appscroll = new JScrollPane(applist);
1084                    apppanel.add(appscroll);
1085                    }
1086                else
1087                    {
1088                    int i = index(appvector, sterm);
1089                    if (i == -1)
1090                        {
1091                        apppanel.remove(appscroll);
1092                        appvector.add(sterm);
1093                        Sort sort = new Sort(appvector);
1094                        appvector = sort.sorted;
1095                        applist = new JList(appvector);
1096                        appscroll = new JScrollPane(applist);
```

```
                    apppanel.add(appscroll);
                    }
                else new Msg("Ce terme a déjà été ajouté.");
                }
            }
        else new Msg("Veuillez sélectionner un terme.");
        apppanel.validate();
        apppanel.repaint();
        }

    /* Enlever un terme de la liste des apparentés */

    else if (e.getActionCommand().equals("fromapp"))
        {
        if (!appvector.isEmpty())
            {
            if (!applist.isSelectionEmpty())
                {
                String sapp = (String)applist.getSelectedValue();
                int i = index(appvector, sapp);
                appvector.removeElementAt(i);
                }
            else new Msg("Veuillez sélectionner un terme apparenté");
            }
        newapp.setText("Nouvel apparenté");
        apppanel.validate();
        apppanel.repaint();
        }

    /* Choisir tous les termes de la liste comme termes apparentés */

    else if (e.getActionCommand().equals("toallapp"))
        {
        apppanel.remove(appscroll);
        appvector = (Vector)vterms.clone();
        applist = new JList(appvector);
        appscroll = new JScrollPane(applist);
        apppanel.add(appscroll);
        apppanel.validate();
        apppanel.repaint();
        }

    /* Supprimer tous les apparentés */

    else if (e.getActionCommand().equals("fromallapp"))
        {
        appvector.clear();
        apppanel.validate();
        apppanel.repaint();
        }

    /* Ajouter un terme de la liste des termes dans la liste des opposés */

    else if (e.getActionCommand().equals("toant"))
        {
        String sterm = (String)termlist.getSelectedValue();

        if (sterm != null)
            {
            if (antvector.isEmpty())
                {
                antpanel.remove(antscroll);
                antvector.add(sterm);
                antlist = new JList(antvector);
                antscroll = new JScrollPane(antlist);
                antpanel.add(antscroll);
                }
            else
                {
                int i = index(antvector, sterm);
                if (i == -1)
                    {
                    antpanel.remove(antscroll);
                    antvector.add(sterm);
                    Sort sort = new Sort(antvector);
                    antvector = sort.sorted;
```

```
1173                        antlist = new JList(antvector);
1174                        antscroll = new JScrollPane(antlist);
1175                        antpanel.add(antscroll);
1176                        }
1177                    else new Msg("Ce terme a déjà été ajouté.");
1178                    }
1179                }
1180            else new Msg("Veuillez sélectionner un terme.");
1181            antpanel.validate();
1182            antpanel.repaint();
1183            }
1184
1185        /* Enlever un terme de la liste des opposés */
1186
1187        else if (e.getActionCommand().equals("fromant"))
1188            {
1189            if (!antvector.isEmpty())
1190                {
1191                if (!antlist.isSelectionEmpty())
1192                    {
1193                    String sant = (String)antlist.getSelectedValue();
1194                    int i = index(antvector, sant);
1195                    antvector.removeElementAt(i);
1196                    }
1197                else new Msg("Veuillez sélectionner un opposé");
1198                }
1199            newant.setText("Nouvel opposé");
1200            antpanel.validate();
1201            antpanel.repaint();
1202            }
1203
1204        /* Choisir tous les termes de la liste comme termes opposés */
1205
1206        else if (e.getActionCommand().equals("toallant"))
1207            {
1208            antpanel.remove(antscroll);
1209            antvector = (Vector)vterms.clone();
1210            antlist = new JList(antvector);
1211            antscroll = new JScrollPane(antlist);
1212            antpanel.add(antscroll);
1213            antpanel.validate();
1214            antpanel.repaint();
1215            }
1216
1217        /* Supprimer tous les opposés */
1218
1219        else if (e.getActionCommand().equals("fromallant"))
1220            {
1221            antvector.clear();
1222            antpanel.validate();
1223            antpanel.repaint();
1224            }
1225
1226        /* Choisir une UA existante comme UA référencée par le nouveau terme */
1227
1228        else if (e.getActionCommand().equals("touas"))
1229            {
1230            int i = uasexlist.getSelectedRow();
1231            if (i == -1)
1232                {
1233                new Msg("Veuillez sélectionner une unité d'apprentissage");
1234                }
1235            else
1236                {
1237                String suaN = (String)uasexlist.getValueAt(i,0);
1238                String suaV = (String)uasexlist.getValueAt(i,1);
1239                Vector v = new Vector();
1240                v.add(suaN);
1241                v.add(suaV);
1242
1243                if (uasrefvector.isEmpty())
1244                        {
1245                        uasrefpanel.remove(uasrefscroll);
1246                        uasrefvector.add(v);
1247                        uasreflist = new noeditjtable(uasrefvector, uasreftitle);
1248                        uasrefscroll = new JScrollPane(uasreflist);
```

```
1249                    uasrefpanel.add(uasrefscroll);
1250                    }
1251            else
1252                    {
1253                    if (uasrefvector.size() == maxuas)
1254                        {
1255                        new Msg("Maximum " + maxuas + " unités
                            d'apprentissage");
1256                        }
1257                    else
1258                        {
1259                        if (!uasrefvector.contains(v))
1260                            {
1261                            uasrefpanel.remove(uasrefscroll);
1262                            uasrefvector.add(v);
1263                            uasreflist = new noeditjtable(uasrefvector,
                            uasreftitle);
1264                            uasrefscroll = new JScrollPane(uasreflist);
1265                            uasrefpanel.add(uasrefscroll);
1266                            }
1267                        else new Msg("Cette unité d'apprentissage a déjà été
                            ajoutée.");
1268                        }
1269                    }
1270            }
1271        uasrefpanel.validate();
1272        uasrefpanel.repaint();
1273        }
1274
1275    /* Supprimer une UA référencée */
1276
1277    else if (e.getActionCommand().equals("fromuas"))
1278        {
1279        if (!uasrefvector.isEmpty())
1280            {
1281            int i = uasreflist.getSelectedRow();
1282            if (i != -1)
1283                {
1284                uasrefvector.removeElementAt(i);
1285                }
1286            else new Msg("Veuillez sélectionner une unité d'apprentissage");
1287            }
1288        uasrefpanel.validate();
1289        uasrefpanel.repaint();
1290        }
1291
1292    /* Ajouter toutes les UAS existantes à la liste des UAS référencées */
1293
1294    else if (e.getActionCommand().equals("toalluas"))
1295        {
1296        if (uasrefvector.size() + uasexvector.size() > maxuas)
1297            {
1298            new Msg("Maximum " + maxuas + " unités d'apprentissage");
1299            }
1300        else
1301            {
1302            uasrefpanel.remove(uasrefscroll);
1303            uasrefvector = (Vector)uasexvector.clone();
1304            uasreflist = new noeditjtable(uasrefvector, uasreftitle);
1305            uasrefscroll = new JScrollPane(uasreflist);
1306            uasrefpanel.add(uasrefscroll);
1307            uasrefpanel.validate();
1308            uasrefpanel.repaint();
1309            }
1310        }
1311
1312    /* Supprimer toutes les UAS référencées */
1313
1314    else if (e.getActionCommand().equals("fromalluas"))
1315        {
1316        uasrefvector.clear();
1317        uasrefpanel.validate();
1318        uasrefpanel.repaint();
1319        }
1320
1321    /* Insérer une image dans la définition */
```

```
1322
1323        else if (e.getActionCommand().equals("insertimage"))
1324            {
1325            if (jfch.showOpenDialog(Teacher.this) != JFileChooser.APPROVE_OPTION)
1326            return;
1327            String path = jfch.getSelectedFile().getPath();
1328            description = path;
1329            imageicon = new ImageIcon(path);
1330            doc = (Document)(kit.createDefaultDocument());
1331            defimg.setEditable(true);
1332            defimg.setDocument(doc);
1333            defimg.insertIcon(imageicon);
1334            defimg.setEditable(false);
1335            repaint();
1336            }
1337
1338        /* Créer un nouveau glossaire */
1339
1340        else if (e.getActionCommand().equals("new"))
1341            {
1342            remove(panel);
1343            paintInterface();
1344            validate();
1345            repaint();
1346            }
1347
1348        /* Charger un glossaire existant */
1349
1350        else if (e.getActionCommand().equals("ex"))
1351            {
1352            new ImportGlossary(this, serv);
1353            }
1354
1355        /* Soumettre la nouvelle traduction */
1356
1357        else if (e.getActionCommand().equals("submit"))
1358            {
1359            String translation = (newtraduc.getText()).trim();
1360
1361            if ((!translation.equalsIgnoreCase("Nouvelle traduction")) &&
            (!translation.equalsIgnoreCase("")))
1362                {
1363                ImageIcon newSelection = (ImageIcon)flag.getSelectedItem();
1364
1365                String language = (String)newSelection.getDescription();
1366
1367                    if (language.startsWith("An"))
1368                        {
1369                        newtraducs[0] = translation;
1370                        }
1371                    else if (language.startsWith("Né"))
1372                        {
1373                        newtraducs[1] = translation;
1374                        }
1375                    else if (language.startsWith("Al"))
1376                        {
1377                        newtraducs[2] = translation;
1378                        }
1379                    else if (language.startsWith("Es"))
1380                        {
1381                        newtraducs[3] = translation;
1382                        }
1383                }
1384            else
1385                {
1386                new Msg("La traduction est invalide.");
1387                }
1388            }
1389
1390        /* Sauver le nouveau terme et toutes ses caractéristiques */
1391
1392        else if (e.getActionCommand().equals("save"))
1393            {
1394            String nt = newterm.getText().trim();
1395            String ab = newabb.getText().trim();
1396            String definition = deftxt.getText();
```

```
1397            String entrad = newtraducs[0];
1398            String dtrad = newtraducs[1];
1399            String gtrad = newtraducs[2];
1400            String estrad = newtraducs[3];
1401
1402            if (containsYorN(vterms, newterm.getText()))
1403            {
1404            new Msg("Ce terme existe déjà.");
1405            newterm.setText("Nouveau terme");
1406            newabb.setText("Abréviation du nouveau terme");
1407            newtraducs[0] = "";
1408            newtraducs[1] = "";
1409            newtraducs[2] = "";
1410            newtraducs[3] = "";
1411            flag.setSelectedIndex(0);
1412            newtraduc.setText("Nouvelle traduction");
1413            newapp.setText("Nouvel apparenté");
1414            newant.setText("Nouvel opposé");
1415            appvector.clear();
1416            antvector.clear();
1417            uasrefvector.clear();
1418            deftxt.setText("");
1419            doc = (Document)(kit.createDefaultDocument());
1420            defimg.setDocument(doc);
1421            imageicon = null;
1422            repaint();
1423            revalidate();
1424            }
1425
1426            else if (!(nt.equals("Nouveau terme")) && !(nt.equals("")) &&
              !(ab.equals("Abréviation du nouveau terme")) &&
              !(definition.equals("")))
1427                {
1428                try
1429                    {
1430                    uas = (Vector)serv.loadUAS(uasrefvector, cours);
1431
1432                    if (imageicon != null)
1433                        {
1434                        defin = new Definition(definition, imageicon);
1435                        }
1436                    else defin = new Definition(definition, null);
1437
1438                    MsgTerm msg = new MsgTerm(this, serv, nt, defin, entrad, dtrad,
                      gtrad, estrad, ab, appvector, antvector, uas);
1439                    }
1440
1441                catch (RemoteException re)
1442                    {
1443                    re.printStackTrace();
1444                    System.out.println("Problème de connexion à distance au
                      serveur");
1445                    }
1446                }
1447            else new Msg("Terme invalide");
1448            }
1449
1450        else if (e.getActionCommand().equals("delete"))
1451            {
1452            if (vterms.size() != 0)
1453                {
1454                if ((int)termlist.getSelectedIndex() != -1)
1455                    {
1456                    String m =
                      changeAp(((String)termlist.getSelectedValue()).trim());
1457                    new MsgDelete(this, serv, m);
1458                    }
1459                else new Msg("Veuillez sélectionner un terme");
1460                }
1461            else new Msg("Le glossaire de ce cours est vide");
1462            }
1463
1464        else if (e.getActionCommand().equals("deleteglossary"))
1465            {
1466            if (vterms.size() != 0)
1467                {
```

```
1468                new MsgDeleteGlossary(this, serv);
1469                }
1470            else new Msg("Le glossaire de ce cours est vide");
1471            }
1472
1473
1474        /* Méthode qui affiche dans le textfield correspondant l'apparenté
        sélectionné */
1475
1476        applist.addMouseListener(new MouseAdapter()
1477        {
1478        public void mouseClicked(MouseEvent e)
1479            {
1480            if (e.getClickCount() >= 1)
1481                {
1482                int index = applist.locationToIndex(e.getPoint());
1483                if (index != -1)
1484                    {
1485                    String appdenom = (String)(appvector.elementAt(index));
1486                    newapp.setText(appdenom);
1487                    }
1488                }
1489            }
1490        });
1491
1492        newapp.addKeyListener(new UpdateList_KeyAdapter(this, applist, appvector,
        newapp));
1493
1494        /* Méthode qui affiche dans le textfield correspondant l'opposé sélectionné
        */
1495
1496        antlist.addMouseListener(new MouseAdapter()
1497        {
1498        public void mouseClicked(MouseEvent e)
1499            {
1500            if (e.getClickCount() >= 1)
1501                {
1502                int index = antlist.locationToIndex(e.getPoint());
1503                if (index != -1)
1504                    {
1505                    String antdenom = (String)(antvector.elementAt(index));
1506                    newant.setText(antdenom);
1507                    }
1508                }
1509            }
1510        });
1511
1512        newant.addKeyListener(new UpdateList_KeyAdapter(this, antlist, antvector,
        newant));
1513        }
1514
1515    }
1516
```

```
1  //Title:        Term
2  //Version:      1
3  //Copyright:    Copyright (c) 2001
4  //Author:       Audrey Lecerf
5  //Company:      MAI PROJECT - FUNDP
6  //Description:  Le Term est l'objet central du glossaire, il est identifié par
                   son nom et le cours auquel y appartient car il existe un
                   glossaire par cours.
7
8
9  import java.awt.*;
10 import javax.swing.*;
11 import java.util.*;
12 import java.io.*;
13
14 class Term implements Serializable
15 {
16 String term;
17 Definition definition;
18 String cours;
19 String englishtranslation = null;
20 String dutchtranslation = null;
21 String germantranslation = null;
22 String spanishtranslation = null;
23 String abbreviation = null;
24 Vector relatedterms = new Vector(); // Vecteur des noms des termes car ces
   termes n'existent pas forcément dans le glossaire
25 Vector antonyms = new Vector(); // Vecteur des noms des termes car ces termes
   n'existent pas forcément dans le glossaire
26 Vector uas = new Vector(); // Vecteurs d'UAS
27
28 public Term(String name, Definition def, String co, String engl, String dutch,
                String germ, String span, String abb, Vector relt, Vector ant,
                Vector ua)
29      {
30      term = name;
31      definition = def;
32      cours = co;
33      englishtranslation = engl;
34      dutchtranslation = dutch;
35      germantranslation = germ;
36      spanishtranslation = span;
37      abbreviation = abb;
38      relatedterms = relt;
39      antonyms = ant;
40      uas = ua;
41      }
42 }
```

```
1  //Title:        TermLoading
2  //Version:      1
3  //Copyright:    Copyright (c) 2001
4  //Author:       Audrey Lecerf
5  //Company:      MAI PROJECT - FUNDP
6  //Description:  Création d'un Term à partir de son identifiant et de son entrée
                   dans la BD
7
8
9  import java.awt.*;
10 import java.lang.Object.*;
11 import javax.swing.*;
12 import java.awt.event.*;
13 import java.util.*;
14 import java.sql.*;
15 import java.sql.Connection.*;
16 import java.rmi.*;
17 import java.io.*;
18
19 import javax.swing.event.*;
20 import javax.swing.border.*;
21
22 public class TermLoading implements Serializable
23 {
24 Term term;
25 String name;
26 String cours;
27 Connection con;
29 ServerRemote server;
29 ImageIcon imageicon = new ImageIcon();
30 ResourceBundle rbConnect;
31 String sDriver;
32 String sURL;
33
34 public TermLoading(String termname, String c, ServerRemote serv) throws
   RemoteException
35     {
36     server = serv;
37     name = termname;
38     cours = c;
39
40     try  // get the PropertyResourceBundle
41         {
42           rbConnect = ResourceBundle.getBundle("TermLoading");
43           sDriver = rbConnect.getString("CSDriver");
44           sURL = rbConnect.getString("CSURL");
45         }
46
47     catch( MissingResourceException mre )
49         {
49         mre.printStackTrace();
50         }
51
52     try
53         {
54         Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
55         con = DriverManager.getConnection("jdbc:odbc:bdmai", "sa", "xxxxxx");
56         DatabaseMetaData dma = con.getMetaData();
57         }
59
59     catch (Exception e)
60         {
61         e.printStackTrace();
62         }
63
64   loadTerm(termname);
65
66     try
67         {
68         con.close();
69         }
70
71     catch(Exception e)
72         {
73         e.printStackTrace();
74         }
```

```
75      }
76
77  public void loadTerm(String s) throws RemoteException
78      {
79      Vector v = new Vector();
80
81      try
82          {
83          v = ((DB)server.giveDB()).getData("SELECT * FROM TERM WHERE NAME = '" +
            s + "' AND COURS = '" + cours + "'", con);
84
85          int sizev = v.size();
86          int x = 0;
87
88          while (x < sizev)
89              {
90              String name = null;
91              String def = null;
92              String cour = null;
93              String engl = null;
94              String dutch = null;
95              String germ = null;
96              String span = null;
97              String abb = null;
98              Vector rlt = new Vector();
99              Vector ant = new Vector();
100             Vector ua = new Vector();
101             String image = null;
102
103             Term t = null;
104
105             name = (String)((Vector)(v.get(x))).get(0);
106
107             cour = (String)((Vector)(v.get(x))).get(1);
108
109             abb = (String)((Vector)(v.get(x))).get(2);
110
111             engl = (String)((Vector)(v.get(x))).get(3);
112
113             dutch = (String)((Vector)(v.get(x))).get(4);
114
115             germ = (String)((Vector)(v.get(x))).get(5);
116
117             span = (String)((Vector)(v.get(x))).get(6);
118
119             def = (String)((Vector)(v.get(x))).get(7);
120
121             image = (String)((Vector)(v.get(x))).get(9);
122
123             Vector vrlt = new Vector();
124             vrlt = ((DB)server.giveDB()).getData("SELECT * FROM RELATEDTERM
                WHERE NAME = '" + s + "' AND COURS = '" + cours + "'", con);
125             int xrlt = 0;
126
127             while (xrlt < vrlt.size())
128                 {
129                 String nrlt = null;
130                 nrlt = (String)((Vector)(vrlt.get(xrlt))).get(2);
131                 rlt.add(nrlt);
132                 xrlt++;
133                 }
134
135             Vector vant = new Vector();
136             vant = ((DB)server.giveDB()).getData("SELECT * FROM ANTONYM WHERE
                NAME = '" + s + "' AND COURS = '" + cours + "'", con);
137             int xant = 0;
138
139             while (xant < vant.size())
140                 {
141                 String nant = null;
142                 nant = (String)((Vector)(vant.get(xant))).get(2);
143                 ant.add(nant);
144                 xant++;
145                 }
146
147             Vector vua = new Vector();
```

```
148             vua = ((DB)server.giveDB()).getData("SELECT A.TITRE, A.PRESENTATION,
                A.STATUS, A.VERSION, A.FACULTE FROM UA A, CHARACTERIZATION C WHERE
                A.TITRE = C.TITRE AND A.VERSION = C.VERSION AND C.NAME = '" + s + "'
                AND C.COURS = '" + cours + "'", con);
149             int xua = 0;
150
151             while (xua < vua.size())
152                 {
153                 String uat = null;
154                 uat = (String)((Vector)(vua.get(xua))).get(0);
155
156                 String uap = null;
157                 uap = (String)((Vector)(vua.get(xua))).get(1);
158
159                 String uas = null;
160                 uas = (String)((Vector)(vua.get(xua))).get(2);
161
162                 String uavstring = (String)((Vector)(vua.get(xua))).get(3);
163                 Integer uavint = Integer.valueOf(uavstring);
164                 int uav = uavint.intValue();
165
166                 String uaf = null;
167                 uaf = (String)((Vector)(vua.get(xua))).get(4);
168
169                 UA newua = new UA(uat, uap, uas, uav, uaf);
170                 ua.add(newua);
171                 xua++;
172                 }
173
174             Statement statimg = null;
175             ResultSet resultimg = null;
176             Blob blob = null;
177
178             statimg = con.createStatement();
179             resultimg = statimg.executeQuery("SELECT Blob_Contenu FROM RESSOURCE
                WHERE DTSIGNATION = '" + image + "'");
180             boolean found = false;
181             while (resultimg.next() && (!found))
182                 {
183                 //blob = resultimg.getBlob("Blob_Contenu");
184                 // ==> A modifier quand on saura pourquoi pas possible de faire
                        getBlob() !
185                 found = true;
186                 }
187
188             if (blob != null)
189                 {
190                 int leng = (int)(blob.length());
191                 imageicon = new ImageIcon(blob.getBytes(1, leng));
192
193                 /* ????????????????? A exploiter ?????????????
194                 File f = new File(image);
195                 FileOutputStream fos = new FileOutputStream(f);
196
197                 InputStream is = blob.getBinaryStream();
198                 System.out.println("is : " + is);
199                 int av = is.available();
200                 byte[] b = new byte[av];
201                 is.read(b);
202
203                 int length = -1;
204
205                 while ((length = is.read(b)) != -1)
206                     {
207                     fos.write(b, 0, length);
208                     }
209
210                 is.close();
211
212                 System.out.println("avant création icon : b = " + b);
213                 imageicon = new ImageIcon(b);
214                 */
215                 System.out.println("après création icon ");
216                 }
217
218             term = new Term(name, new Definition(def, imageicon), cour, engl,
```

```
                dutch, germ, span, abb, rlt, ant, ua);
219
220             x++;
221             }
222         }
223
224     catch (Exception e)
225         {
226         e.printStackTrace();
227         }
228     }
229 }
```

```
1  //Title:        UA
2  //Version:      1
3  //Copyright:    Copyright (c) 2001
4  //Author:       Audrey Lecerf
5  //Company:      MAI PROJECT - FUNDP
6  //Description:  Une UA est identifiée par son titre, sa version et est
                   caractérisée par sa présentation, son statut et sa présentation
7
8
9  import java.awt.*;
10 import javax.swing.*;
11 import java.util.*;
12 import java.io.*;
13
14 class UA implements Serializable
15 {
16 String titre = null;
17 String presentation = null;
18 String statut = null;
19 int version;
20 String fac = null;
21
22 public UA(String t, String p, String s, int v, String f)
23     {
24     titre = t;
25     presentation = p;
26     statut = s;
27     version = v;
28     fac = f;
29     }
30 }
```

```
1  //Title:        UALoading
2  //Version:      1
3  //Copyright:    Copyright (c) 2001
4  //Author:       Audrey Lecerf
5  //Company:      MAI PROJECT - FUNDP
6  //Description:  Création d'une UA à partir de son identifiant et de son entrée
                   dans la BD
7
8
9  import java.awt.*;
10 import java.lang.Object.*;
11 import javax.swing.*;
12 import java.awt.event.*;
13 import java.util.*;
14 import java.sql.*;
15 import java.sql.Connection.*;
16 import java.rmi.*;
17 import java.io.*;
18
19 import javax.swing.event.*;
20 import javax.swing.border.*;
21
22 public class UALoading implements Serializable
23 {
24 UA ua;
25 Connection con;
26 ServerRemote server;
27
28 public UALoading(String uaname, int uaversion, ServerRemote serv, Connection c)
   throws RemoteException
29     {
30     server = serv;
31     con = c;
32     loadUA(uaname, uaversion, server);
33     }
34
35 public void loadUA(String s, int i, ServerRemote server) throws RemoteException
36     {
37     Vector v = new Vector();
38
39     try
40         {
41         v = ((DB)server.giveDB()).getData("SELECT * FROM UA WHERE TITRE = '" + s
           + "' AND VERSION = " + i, con);
42
43         int sizev = v.size();
44         int x = 0;
45
46         while (x < sizev)
47             {
48             String titre = null;
49             int version;
50             String presentation = null;
51             String statut = null;
52             String faculte = null;
53
54             titre = (String)((Vector)(v.get(x))).get(0);
55
56             presentation = (String)((Vector)(v.get(x))).get(1);
57
58             statut = (String)((Vector)(v.get(x))).get(3);
59
60             version =
               (int)Integer.parseInt((String)((Vector)(v.get(x))).get(4));
61
62             faculte = (String)((Vector)(v.get(x))).get(5);
63
64             ua = new UA(titre, presentation, statut, version, faculte);
65
66             x++;
67             }
68
69         }
70
71      catch (Exception e)
72         {
```

```
73              e.printStackTrace();
74              }
75          }
76      }
77
78
```

```
1  //Title:        UpdateList_KeyAdapter
2  //Version:      1
3  //Copyright:    Copyright (c) 2001
4  //Author:       Audrey Lecerf
5  //Company:      MAI PROJECT - FUNDP
6  //Description:  Défilement d'une liste à partir des lettres entrées dans un
                   TextField
7
8
9  import java.awt.*;
10 import java.awt.event.*;
11
12 import javax.swing.*;
13
14 import java.util.*;
15
16 class UpdateList_KeyAdapter extends KeyAdapter
17 {
18   Student std;
19   Teacher tch;
20   JList jl;
21   Vector v = new Vector();
22   JTextField jtf;
23
24   UpdateList_KeyAdapter(Student st, JList l)
25     {
26     this.std = st;
27     this.jl = l;
28     }
29
30   UpdateList_KeyAdapter(Teacher t, JList l, Vector vector, JTextField tf)
31     {
32     this.tch = t;
33     this.jl = l;
34     this.v = vector;
35     this.jtf = tf;
36     }
37
38   public void keyPressed(java.awt.event.KeyEvent e)
39     {
40     if (tch == null)
41         {
42         std.selectedterm_keyPressed(e, jl);
43         }
44     else if (std == null)
45         {
46         tch.new_keyPressed(e, jl, v, jtf);
47         }
48     }
49 }
```